



Universidad  
Carlos III de Madrid

Departamento de Informática

PROYECTO FIN DE CARRERA

# Diseño e implementación de una aplicación Android para recordatorios

Autor: Óscar Magro Segura

Tutor: Alejandro Calderón Mateos

Leganés, 3 de Octubre de 2011



Título: Diseño e implementación de una aplicación Android para recordatorios  
Autor: Óscar Magro Segura  
Director: Alejandro Calderón Mateos

## EL TRIBUNAL

Presidente: David Expósito Singh

Vocal: Ana Isabel González-Tablas Ferreres

Secretario: Oscar Pérez Alonso

Realizado el acto de defensa y lectura del Proyecto Fin de Carrera el día 3 de Octubre de 2011 en Leganés, en la Escuela Politécnica Superior de la Universidad Carlos III de Madrid, acuerda otorgarle la CALIFICACIÓN de

VOCAL

SECRETARIO

PRESIDENTE



# Agradecimientos

Quería agradecer a mis padres, Mariano y Paloma, por un apoyo incondicional. Por estar siempre a mi lado, por darme todo y no pedir nada a cambio. Por tenderme la mano cada vez que caía y por hacerme como soy. Nunca podré agradecerles lo suficiente todo lo que han hecho por mí tanto a lo largo de la carrera como fuera de ella, ni sobre todo, por el regalo más preciado que me han dado, la Vida, pero aun así gracias de todo corazón.

A mi hermana pequeña Marta, que pese a nuestras pequeñas disputas siempre está ahí para recordarme que dos son mejor que uno y por animarme sin ella saberlo en los momentos de soledad. Por cuidar de mí cuando creí que era yo quien cuidaba de ella. Por esto y por mucho más, gracias.

A mi abuela Purificación, que siempre nos está cuidando. Por compartir con nosotros su experiencia y ayudarnos a no cometer los mismos errores. Por educarnos y hacerse cargo de nosotros cuando nuestros padres no podían.

A Laura, una persona que ha puesto patas arriba mi vida, pero que siempre me ha apoyado en mis decisiones y me ha ayudado en los momentos más difíciles. Por escucharme y ayudarme a ir hacia delante y que nunca me ha dejado darme por vencido. Por haberme animado a seguir con el proyecto y por hacer mucho más fáciles los años que llevamos juntos.

A Vicente, por seguir a mi lado incluso cuando no lo merecía. Por apoyarme y animarme a seguir con este trabajo. Por todos los consejos dados y por compartir sus conocimientos, pero sobre todo, por enseñarme un camino y ayudarme a andarlo.

A Álvaro, Roberto y Carlos, tres amigos que me siguen desde la infancia y con los cuales siempre he podido contar y nunca me han dejado de lado. Por todas las historias vividas juntos y por hacer que todo ese tiempo haya pasado con una buena dosis de humor, alegría y buenos momentos.

A Juanma, por hacer divertido lo aburrido. Por las risas durante los desayunos, prácticas y horas de trabajo. Por su conocimiento y la inestimable ayuda a lo largo de este proyecto así como durante la carrera.

Al laboratorio de informática por hacer las mañanas más divertidas y amenizar el desayuno en la cafetería.

A Iván y David, dos amigos y compañeros que siguen a mi lado y de los que siempre tendré su apoyo.

A mi tutor, Alejandro Calderón, por su ayuda a lo largo del proyecto y carrera. Por convertirse en uno de los profesores de referencia y por que sin él, el desarrollo de este trabajo no habría sido posible.

A mis compañeros de universidad Javi Roca, por todas las prácticas que hemos pasado desde el inicio, por sudarlas y reírlas juntos. Christian, por estar siempre dispuesto a ayudar y por dejar siempre su último comentario, Lidia, por sus ánimos, Alberto, por equilibrar el grupo siempre con su bondad, Iván, por ofrecer su humor ácido, Lissar, por compartir su conocimiento, Antonio, por esas partidas en la cafetería y por compartir el estrés de las prácticas,... y a todos aquellos a los que me dejo en la lista pero no los olvido.

Por último quería agradecer y dedicar este proyecto a cuatro personas muy importantes para mí y que ya no están conmigo. A mi abuela Cándida, a mi abuelo Mariano, a mi abuelo Joaquín y a mi tía Mercedes. Me habría gustado haber podido compartir este momento con ellos y darles las gracias por todo lo que han hecho por mí. ¡No os olvido!



# Resumen

El proyecto llevado a cabo muestra el desarrollo de una aplicación para un dispositivo móvil para la plataforma Android. Para ello es necesario estudiar tanto el mercado actual para dispositivos y aplicaciones, así como esta plataforma para conocer sus principales características y el funcionamiento de su arquitectura para así comprender como se debe desarrollar en ella. También se realizará un análisis de los requisitos necesarios para realizar la el diseño y desarrollo de la aplicación. En esta documentación se recogen todos los pasos seguidos para completar el proyecto.

**Palabras clave:** Android, aplicación, desarrollo, análisis, diseño, requisitos, implementación.





# Abstract

The project carried on shows the development of an application for a mobile device for the Android platform. For this it is necessary to consider the current market devices and applications as well as this platform, in order to learn about its main features and how its architecture works just to understand how to develop on it. Also it will be performed an analysis of the requirements for the design and the application development. This documentation contains all the steps followed to complete the project.

**Keywords:** Android, application, development, design, analysis, requirement, implementation.

# Índice general

<b>1. INTRODUCCIÓN Y OBJETIVOS .....</b>	<b>1</b>
1.1 Introducción .....	1
1.2 Objetivos .....	2
1.3 Fases del desarrollo .....	2
1.4 Medios empleados.....	3
1.5 Estructura de la memoria .....	3
<b>2. ESTADO DE LA CUESTIÓN.....</b>	<b>5</b>
2.1 Introducción .....	5
2.2 Estudio de las plataformas móviles.....	7
<b>3. TRABAJANDO CON ANDROID Y EL CALENDARIO DE GOOGLE.....</b>	<b>14</b>
3.1 Introducción .....	14
3.2 Preparar el entorno .....	15
3.3 Uso de Librerías externas.....	18
3.4 Interactuando con servidores de correo.....	20
3.5 Uso de los Servicios del Calendario de Google .....	23
3.5.1 Autenticación para servicios de Google.....	23
3.5.2 Listas de calendarios en Google Calendar .....	27
3.5.3 Eventos en Google Calendar .....	29
3.6 Almacenamiento de los datos.....	32
<b>4. ANÁLISIS, DISEÑO E IMPLEMENTACIÓN DE “REBRAINDME” .....</b>	<b>33</b>
4.1 Introducción .....	33
4.2 Análisis.....	34
4.2.1 Requisitos de Usuario .....	34
4.3 Diseño .....	55
4.3.1 Diseño de la Arquitectura.....	55
4.3.2 Diseño de la Base de Datos .....	62
4.3.3 Diseño de la Interfaz .....	63
4.4 Implementación.....	73
4.4.1 Base de datos.....	73
4.4.2 Gestor de correo .....	77

4.4.3 Gestor de Eventos .....	80
4.4.4 Gestor de Notas .....	85
4.4.5 Interfaz de Usuario .....	88
<b>5. PLANIFICACIÓN Y PRESUPUESTO .....</b>	<b>92</b>
5.1 Planificación.....	92
5.2 Presupuesto .....	94
<b>6. CONCLUSIONES Y LÍNEAS FUTURAS.....</b>	<b>97</b>
6.1 Conclusiones .....	97
6.2 Líneas futuras .....	99
<b>7. PLATAFORMA ANDROID .....</b>	<b>101</b>
a.1. Introducción.....	101
a.2. Versiones .....	102
a.3. Arquitectura .....	105
a.3.1. Nivel de aplicaciones .....	105
a.3.2. Marco de trabajo de aplicaciones .....	105
a.3.3. Librerías de Android .....	106
a.3.4. Runtime de Android .....	106
a.3.5. Kernel de Linux.....	106
a.4. Aplicaciones .....	107
a.5. Proveedores de contenido.....	111
a.6. Interfaz de Usuario .....	112
a.7. Recursos de Aplicación .....	114
a.8. Seguridad y permisos .....	115
a.9. Archivo AndroidManifest.xml .....	116
<b>8. GLOSARIO .....</b>	<b>117</b>
<b>9. REFERENCIAS.....</b>	<b>118</b>

# Índice de figuras

Figura 1. Evolución de las líneas móviles.....	6
Figura 2. Motorola DynaTAC 800x.....	6
Figura 3. Teléfonos móviles actuales.....	6
Figura 4. Smartphones .....	7
Figura 5. Encuesta acceso a noticias por diferentes medios[5].....	7
Figura 6. Interfaces de Symbian^3, IOS 4.0, BlackBerry 6.0, Android 2.3 y Windows Phone 7.....	9
Figura 7. Cuota de mercado de las plataformas móviles .....	9
Figura 8. Estudio de aplicaciones disponibles en app stores .....	10
Figura 9. Instalar nuevo software en Eclipse .....	15
Figura 10. Descarga e instalación del plug-in .....	16
Figura 11. Configurar plug-in .....	16
Figura 12. Referenciar SDK.....	17
Figura 13. Descarga de nuevos paquetes .....	17
Figura 14. Librerías necesarias para GoogleCalendar .....	19
Figura 15. Envío de correo.....	20
Figura 16. Autenticación OAuht .....	24
Figura 17. Autenticación ClientLogin.....	25
Figura 18. Panel de Control Cuentas y Sincronización.....	26
Figura 19. Código Atom para crear calendarios .....	27
Figura 20. Sentencia de edición en Atom .....	28
Figura 21. Sentencia de edición en Json-C .....	28
Figura 22. Modificar un calendario en Atom.....	28
Figura 23. Crear Evento web Calendario Google .....	29
Figura 24. Atributos de un evento.....	30
Figura 25. Creación de un Evento en Atom .....	30
Figura 26. Modificar un evento en Atom.....	31
Figura 27. Caso de Uso: Configuración de la aplicación.....	44
Figura 28. Caso de uso: Gestión de correo .....	46
Figura 29. Caso de uso: Gestión de eventos .....	50
Figura 30. Caso de uso: Gestión de notas .....	52

Figura 31. Componentes de la aplicación .....	55
Figura 32. Diagrama de Componentes .....	56
Figura 33. Tablas Base de Datos .....	62
Figura 34. Inicio Aplicación Menú Principal .....	63
Figura 35. Interfaz de configuración .....	64
Figura 36. Interfaces emergentes para el correo .....	65
Figura 37. Interfaces Bandeja - Correo .....	66
Figura 38. Añadir Correo al Calendario .....	67
Figura 39. Enviar correo electrónico .....	68
Figura 40. Interfaces Añadir Nota Contacto .....	69
Figura 41. Interfaces Visualizar Notas .....	70
Figura 42. Interfaces Visualizar Eventos .....	71
Figura 43. Interfaz Salida Aplicación .....	72
Figura 44. Sentencias Creación Tablas .....	73
Figura 45. Ejecución Sentencias SQL .....	74
Figura 46. Insertar Datos Tabla .....	75
Figura 47. Eliminar entradas .....	75
Figura 48. Consulta Tabla .....	75
Figura 49. Modificar entrada .....	76
Figura 50. Configurar Conexión IMAP .....	77
Figura 51. Conexión y obtención de mensajes .....	78
Figura 52. Conexión SMTP .....	78
Figura 53. Enviar Correo .....	79
Figura 54. Autenticación Google Calendar .....	80
Figura 55. Creación URL calendario del usuario .....	81
Figura 56. Ejemplo Evento Calendario Google .....	81
Figura 57. Creación de un Evento en Java .....	82
Figura 58. Creación Evento Android .....	82
Figura 59. Enviar un Evento a GoogleCalendar .....	83
Figura 60. Captura excepción de redirección url .....	84
Figura 61. Listar contactos usuario .....	85
Figura 62. Insertar nueva nota en el sistema .....	86
Figura 63. Mostrar una nota .....	86
Figura 64. Eliminar nota del sistema .....	87
Figura 65. XML Lista de elementos .....	88
Figura 66. Rellenar una lista .....	89
Figura 67. Código interfaz de creación de correo .....	90
Figura 68. Crear ventana emergente .....	90
Figura 69. Capturar tecla Back .....	91
Figura 70. Diagrama de Gantt respecto a la Planificación del Proyecto .....	93
Figura 71. Arquitectura de Android .....	105
Figura 72. Ciclo de vida de una actividad .....	109
Figura 73. Ciclos de vida de un servicio .....	110
Figura 74. Jerarquía de interfaz de usuario .....	112
Figura 75. Código de ejemplo de una interfaz de usuario .....	113
Figura 76. Salida de ejemplo de una interfaz de usuario .....	113
Figura 77. Uso de recursos alternativos .....	114
Figura 78. Permiso acceso internet .....	115
Figura 79. Estructura general del archivo AndroidManifest.xml .....	116

# Índice de tablas

Tabla 1. Comparativa Android 2.2 Vs IOS 4 .....	12
Tabla 2. Comparativa entre POP e IMAP .....	21
Tabla 3. Requisito de Capacidad 1 .....	35
Tabla 4. Requisito de Capacidad 2 .....	35
Tabla 5. Requisito de Capacidad 3 .....	35
Tabla 6. Requisito de Capacidad 4 .....	36
Tabla 7. Requisito de Capacidad 5 .....	36
Tabla 8. Requisito de Capacidad 6 .....	36
Tabla 9. Requisito de Capacidad 7 .....	37
Tabla 10. Requisito de Capacidad 8 .....	37
Tabla 11. Requisito de Capacidad 9 .....	38
Tabla 12. Requisito de Capacidad 10 .....	38
Tabla 13. Requisito de Capacidad 11 .....	38
Tabla 14. Requisito de Capacidad 12 .....	39
Tabla 15. Requisito de Capacidad 13 .....	39
Tabla 16. Requisito de Capacidad 14 .....	39
Tabla 17. Requisito de Capacidad 15 .....	40
Tabla 18. Requisito de Capacidad 16 .....	40
Tabla 19. Requisito de Capacidad 17 .....	40
Tabla 20. Requisito de restricción 1 .....	41
Tabla 21. Requisito de restricción 2 .....	41
Tabla 22. Requisito de restricción 3 .....	41
Tabla 23. Requisito de restricción 4 .....	42
Tabla 24. Requisito de restricción 5 .....	42
Tabla 25. Caso de uso CU 01 .....	44
Tabla 26. Caso de uso CU 02 .....	45
Tabla 27. Caso de uso CU 03 .....	46
Tabla 28. Caso de uso CU 04 .....	47
Tabla 29. Caso de uso CU 05 .....	47

Tabla 30. Caso de uso CU 06.....	48
Tabla 31. Caso de uso CU 07.....	48
Tabla 32. Caso de uso CU 08.....	49
Tabla 33. Caso de uso CU 09.....	50
Tabla 34. Caso de uso CU 10.....	51
Tabla 35. Caso de uso CU 11.....	52
Tabla 36. Caso de uso CU 12.....	52
Tabla 37. Caso de uso CU 13.....	53
Tabla 38. Caso de uso CU 14.....	54
Tabla 39. Componente MailConnector Manager.....	57
Tabla 40. Componente Mail Manager .....	57
Tabla 41. Componente CalendarConnector Manager.....	58
Tabla 42. Componente Event Manager.....	58
Tabla 43. Componente Event Mail Service .....	59
Tabla 44. Componente Contact Service.....	59
Tabla 45. Componente Note Manager .....	59
Tabla 46. Componente Settings .....	60
Tabla 47. Matriz Trazabilidad Requisitos-Componentes.....	61
Tabla 48. Planificación de Tareas .....	92
Tabla 49. Duración real por tarea.....	95
Tabla 50. Costes de amortización .....	96
Tabla 51. Resumen presupuesto.....	96



# Capítulo 1

## Introducción y objetivos

### 1.1 Introducción

El presente documento recoge toda la información del desarrollo llevado a cabo para el proyecto final de carrera. Este proyecto surge a partir de la motivación de desarrollar una aplicación para una plataforma móvil que permita al usuario poder recordar citas y eventos cumplimentando al organizador del propio dispositivo y a su vez permitirle acceder a la herramienta más utilizada que es el correo electrónico, la cual será el núcleo de esta aplicación.

Con este objetivo global, se ha utilizado para el desarrollo la plataforma Android al ser una de las plataformas con mayor ratio de crecimiento del mercado y por permitir mediante interfaces, crear todas las funcionalidades que eran necesarias para el desarrollo de esta aplicación como se podrá ver a lo largo del documento.

## 1.2 Objetivos

El objetivo principal de este proyecto es desarrollar una aplicación para un dispositivo móvil que facilite al usuario la organización de su tiempo permitiéndole acceder a su correo electrónico y poder gestionar sus citas mediante eventos dentro de uno de los mayores servicios más utilizados que ofrece Google, que es su aplicación Google Calendar. A estas funcionalidades se le suma una extensión que es permitir almacenar unos recordatorios en forma de notas a los contactos que posee el usuario dentro de su dispositivo.

Por lo tanto, mediante estos objetivos se consigue:

- ❖ Analizar las distintas plataformas móviles del actual mercado, conociendo su crecimiento y las funcionalidades que ofrecen así como sus limitaciones.
- ❖ Estudiar en profundidad la plataforma Android conociendo su arquitectura y los conceptos necesarios para el desarrollo de sus aplicaciones.
- ❖ Ampliar conocimientos de lenguajes de programación como Java.
- ❖ Aprender a realizar proyectos de mayor envergadura que los realizados durante la carrera.

## 1.3 Fases del desarrollo

El desarrollo de este proyecto se ha dividido en siete fases. La primera de ellas consiste en estudiar el mercado de plataformas existentes y conocer las distintas aplicaciones existentes así como la viabilidad del desarrollo de esta aplicación. Posteriormente, al decidirse por un desarrollo para la plataforma Android, ha sido necesario estudiar su arquitectura y funcionamiento así como los conceptos necesarios para realizar desarrollos de aplicaciones en la misma.

Con los conceptos claros, la siguiente fase era realizar un análisis de las funcionalidades que requería la aplicación y estudiar sus requisitos. Con esta información, se procedió a diseñar la arquitectura de la misma analizando los componentes necesarios que debían desarrollarse.

Teniendo las fases anteriores completadas, se tenía ya toda la información necesaria para empezar con la implementación de la aplicación. Para ello se siguieron las pautas proporcionadas por las mismas para evitar desviarse de los objetivos principales definidos.

Por último, dentro del desarrollo, quedó realizar una batería de pruebas para comprobar el correcto comportamiento de la aplicación y corrigiendo los errores que iban apareciendo durante las misas.

Terminando todo el proceso de desarrollo, la última fase ha sido la elaboración de la documentación de todo el proyecto.

## 1.4 Medios empleados

Para llevar a cabo el desarrollo de este proyecto se ha sido necesario el uso de los siguientes elementos de software:

- ❖ Un sistema operativo: Windows Professional 7 de 64 bits
- ❖ Herramienta de desarrollo: IDE Eclipse 3.6 con las herramientas de Android en el plug-in ADT rev.11.
- ❖ Librerías de desarrollo: Java SDK 6, Android SDK rev.11, JavaMail para Android, Api cliente de Google.
- ❖ Microsoft Office 2010 y Microsoft Project 2010.

Por lo que a equipo se refiere, para la implementación de la aplicación ha sido necesario un portátil con todos los elementos software descritos anteriormente instalados en él.

## 1.5 Estructura de la memoria

A lo largo de este documento el lector se encontrará con diferentes capítulos que detallan las fases del desarrollo de este proyecto. A continuación se resumen brevemente para tener una idea global de lo que contendrá cada uno.

- ❖ Capítulo 1. Introducción. Introducción y Objetivos: Contiene un pequeño resumen del proyecto relatando la motivación que ha producido este desarrollo así como los objetivos del mismo, las fases de desarrollo y los medios necesarios para su realización. También incluye una visión global del documento.
- ❖ Capítulo 2. Estado de la cuestión: Este apartado comprende un análisis actual de las plataformas del mercado analizando las más idóneas para desarrollar aplicaciones software.

- ❖ Capítulo 3. Trabajando con Android y el Calendario de Google: En este capítulo se describen los más importantes de Android y cómo crear un entorno de trabajo. Así mismo se explica cómo se debe trabajar con servidores de correo y cómo interactuar con los servicios del calendario de Google.
- ❖ Capítulo 4. Implementación: Se encontrarán las fases de análisis, diseño e implementación de la aplicación que se han seguido para su desarrollo.
- ❖ Capítulo 5. Planificación y Presupuesto: Contiene una planificación realizada del desarrollo del proyecto y se establece un presupuesto en base a dicha planificación.
- ❖ Capítulo 6. Conclusiones y Líneas Futuras: En este capítulo se describirán las conclusiones obtenidas tras el desarrollo del proyecto así como unas líneas futuras de desarrollo e investigación que podrán realizarse a partir de este proyecto como ampliación del mismo.
- ❖ Apéndice A. Plataforma Android: Contiene un detallado resumen de la plataforma Android explicando las principales características tales como su arquitectura. Es una lectura recomendada si no se tienen nociones de esta plataforma.
- ❖ Glosario: En esta sección el lector podrá revisar los términos más técnicos utilizados en este documento, de manera que le resulte más fácil su lectura.
- ❖ Referencias: Contiene las referencias utilizadas tanto para el desarrollo de la aplicación como para la elaboración de este documento.

# Capítulo 2

## Estado de la cuestión

### 2.1 Introducción

Antes de comentar los diferentes dispositivos móviles existentes, es necesario dar una definición que los describa en detalle. Esta definición podría ser “*aparato de pequeño tamaño, con algunas capacidades de procesamiento, móviles o no, con conexión permanente o intermitente a una red, con memoria limitada, diseñados específicamente para una función pero que pueden llevar a cabo otras funciones más generales*”. [3]

Se puede encontrar una gran variedad de diferentes dispositivos en el mercado, como pueden ser PDAs, teléfonos móviles, portátiles, lectores de libros electrónicos, tabletas digitales, etc.

De entre todos ellos, los teléfonos móviles han llegado a convertirse en una parte esencial en nuestras vidas. Siguiendo los estudios realizados por la Comisión del Mercado de las Telecomunicaciones (CMT)[4], se puede ver el crecimiento de estos dispositivos en la sociedad. El siguiente gráfico muestra la evolución de las líneas móviles desde el año 2000 hasta el 2010 en España, viendo como el crecimiento de éstas ha dado un gran salto en los últimos años.

## Evolución de las líneas de móviles

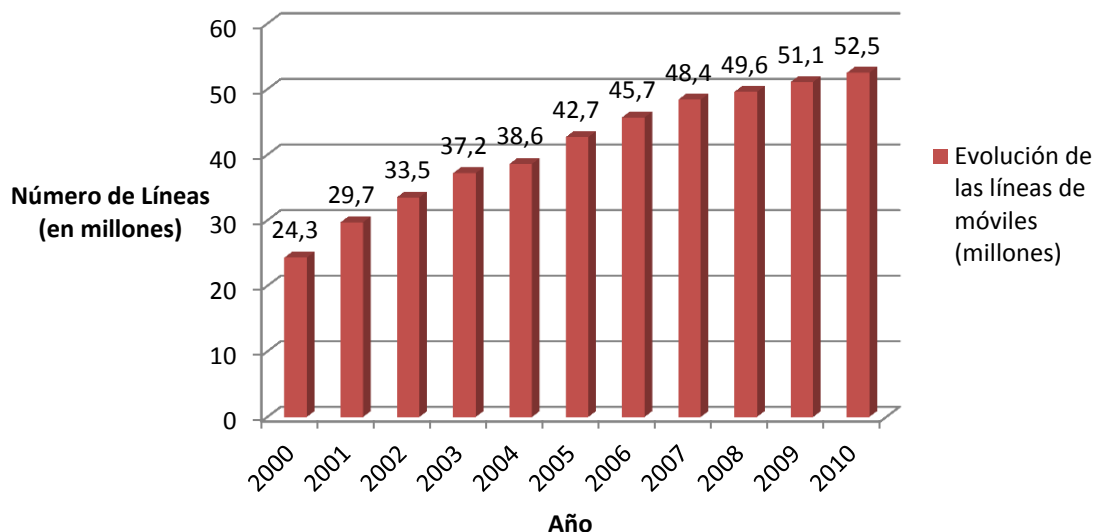


Figura 1. Evolución de las líneas móviles

El crecimiento de este sector viene dado, entre otras causas, por el avance tecnológico. Este avance ha permitido que los teléfonos móviles ofrezcan una serie de funcionalidades que hace unos años eran impensables. El primer teléfono móvil del mercado, desarrollado por Motorola a principio de los 80, fue el DynaTac 8000x cuya única funcionalidad era poder realizar llamadas telefónicas. Con los años y la creación de un software más amigable ha permitido que los teléfonos móviles adquieran funciones complementarias como enviar mensajes de texto, reproducir contenido multimedia, conexión a Internet y la posibilidad de incluir funciones GPS entre otras.



Figura 2. Motorola DynaTAC 800x



Figura 3. Teléfonos móviles actuales

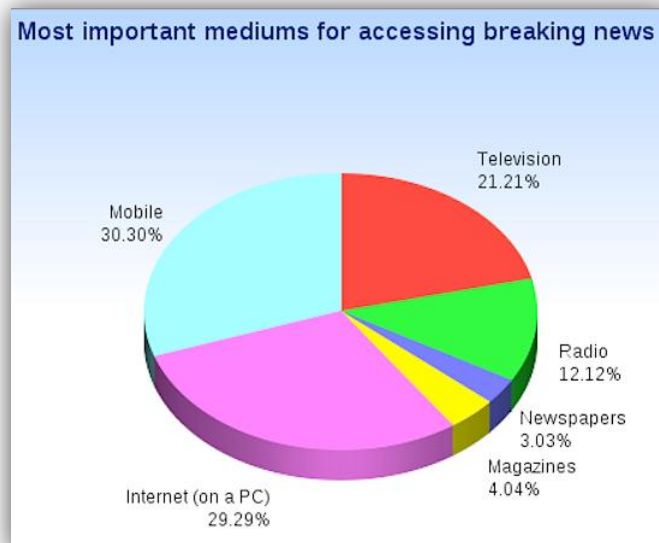
Pero este avance no sólo implica aumentar y mejorar funcionalidades como las mencionadas, si no que las características técnicas de los terminales se optimizan y cambian para poder proporcionar al usuario nuevas ventajas y comodidades. El aumento de la capacidad de procesamiento, la resolución de las pantallas, uso de cámaras de fotos o de video o la incorporación de pantallas táctiles son alguno de estos ejemplos.

Estas tendencias no hacen otra cosa si no que dejar atrás al terminal móvil de toda la vida suplantándolo por dispositivos que son capaces de funcionar con sistemas operativos que permiten la integración de software ofreciendo las mismas funcionalidades que un ordenador pero a un nivel más bajo de rendimiento. De esta manera aparece el término “Smartphone” o teléfono inteligente. Estos dispositivos permiten soportar aplicaciones totalmente personalizables aumentando sus funcionalidades hasta extremos casi ilimitados. Dichas tendencias se reflejan en el estudio de la consultora IDC que indica que ya se venden más smartphones que teléfonos tradicionales[21] habiendo aumentado la venta de éstos un 48% entre el segundo trimestre de 2010 y el segundo trimestre de 2011, mientras que las de móviles han caído el 29% en el mismo periodo.



**Figura 4. Smartphones**

Su capacidad de conexión a internet y el completo soporte al correo electrónico los hacen ser prácticamente indispensables ya que cada día más gente utiliza estos dispositivos para permanecer conectados al mundo mediante redes sociales, correo electrónico y noticias, de tal manera que el uso de ordenadores, televisiones u otros dispositivos están pasando a un segundo plano. Este crecimiento de mercado incita a pensar que este sector está en pleno auge y desarrollar para él puede ser una idea muy interesante. Para ello se analizarán más adelante las diferentes plataformas de desarrollo.



**Figura 5. Encuesta acceso a noticias por diferentes medios[7]**

## 2.2 Estudio de las plataformas móviles

Si se desea desarrollar para un dispositivo móvil es necesario conocer las plataformas de desarrollo existentes en el mercado. Actualmente las plataformas más populares en los terminales móviles son Symbian, IOS, Android, RIM y Windows Mobile, aunque existen otras pero mucho menos utilizadas. A continuación se describirán brevemente cada una de ellas, lo cual permitirá tener un conocimiento básico de las diferentes plataformas.

Symbian apareció con la alianza de varias empresas de telefonía móvil siendo algunos ejemplos Nokia, Sony Ericsson, Siemens y Samsung. El objetivo fue crear un sistema operativo para terminales móviles que pudiera competir con el de Palm o Windows Mobile de Microsoft. La versión más actual es Symbian^3 lanzada en Septiembre de 2010 y diseñada para la nueva generación de móviles introduciendo nuevas características como nuevas arquitecturas gráficas de 2D y 3D, así como mejoras en la interfaz de usuarios.

IOS, en cambio, es un sistema operativo perteneciente a Apple, derivado de Mac OS x basado en Darwin BSD. Originalmente fue desarrollado para iPhone, aunque actualmente se ha extendió a otros dispositivos Apple como pueden ser el iPod Touch o el iPad. La versión estable más alta es la IOS 4.2.1 lanzada en Noviembre de 2010 aunque en fase beta se encuentra actualmente la versión 4.3 que apareció en Febrero de este mismo año 2011.

Research In Motion, o RIM, dispone de un sistema operativo BlackBerry S.O. el cual como su nombre indica está claramente orientado a la línea de smartphones de BlackBerry. El núcleo está basado en Java y la versión estable más actual es la 6.0 que apareció en Octubre de 2010. Este sistema operativo está orientado a un uso más profesional como gestor de correo y agenda.

El sistema operativo Android está basado en Linux y está pensado para dispositivos móviles como smartphones y tablets. Inicialmente fue desarrollado por Android Inc. que fue comprada por Google en el año 2005 y fue anunciado en Noviembre de 2007. Actualmente la última versión estable de la cual se dispone en el mercado es la 2.3, aunque no está disponible para todos los terminales y todas las compañías.

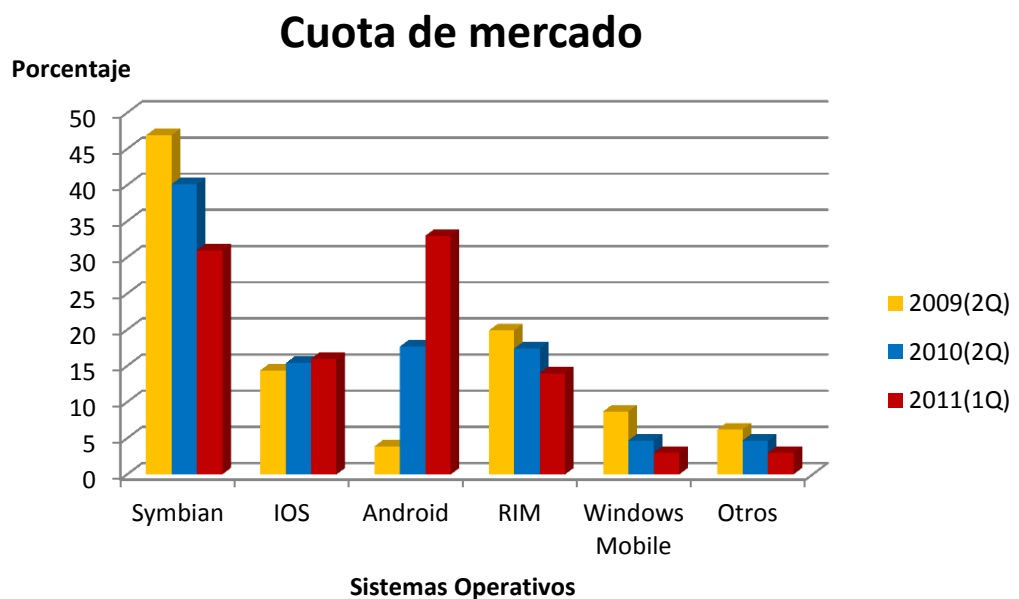


La última plataforma es Windows Mobile, que como es de esperar ha sido desarrollada por Microsoft pensado para plataformas móviles. Se basa en el núcleo del sistema operativo Windows CE y cuenta con un conjunto de aplicaciones básicas utilizando las API de Microsoft Windows y estéticamente está diseñado para ser similar a las versiones de escritorio de Windows. La última versión de esta plataforma es Windows Phone 7 que apareció en Febrero de 2010.



**Figura 6. Interfaces de Symbian^3, IOS 4.0, BlackBerry 6.0, Android 2.3 y Windows Phone 7**

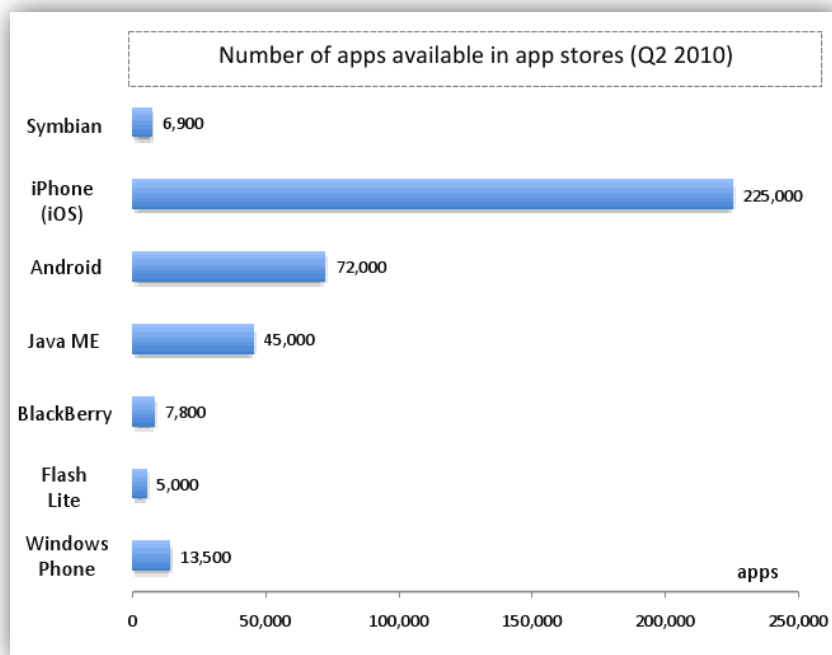
Teniendo ya una pequeña idea de las plataformas actuales, es interesante mirar el siguiente gráfico en el cual se muestran las diferentes cuotas de mercado en todo el mundo a lo largo de los años 2009, 2010 e inicios del 2011 de las mismas. Estos datos fueron tomados en los segundos cuartos anuales de 2009 y 2010 y en el inicio del año 2011.



**Figura 7. Cuota de mercado de las plataformas móviles**

Como se puede observar, de las cinco plataformas, Symbian, RIM y Windows Mobile han perdido cuota de mercado a lo largo de estos años. La más afectada ha sido Symbian al perder un total de 15% de cuota, siendo una cifra bastante significativa. Este terreno cedido por las tres plataformas ha sido tomado por Android e IOS que han aumentado su proporción, especialmente Android, llegando a conseguir casi un 30% más desde el 2009.

Estos datos ya de por sí son bastante reveladores y dan una cierta idea de cuáles son las plataformas más interesantes para desarrollar, pero aún se puede investigar algo más a cerca de estas plataformas. Otra cuestión que es importante tomar en cuenta es el nivel de desarrollo existente en cada plataforma así como su actividad en el número de productos y compra de los mismos. El siguiente gráfico muestra un estudio aproximado del número de aplicaciones disponibles para cada plataforma.



**Figura 8. Estudio de aplicaciones disponibles en app stores**

Este estudio fue tomado en el segundo cuatrimestre de 2010 analizando las diferentes tiendas para las distintas plataformas móviles. De todas ellas, la plataforma IOS para iPhone posee el mayor número de aplicaciones con más de 200.000, seguida de Android superando las 70.000. Con estos datos se puede llegar a la conclusión que tanto Android como IOS iPhone poseen una mayor actividad de desarrollo comparada con el resto de plataformas.

Si tenemos en cuenta los datos correspondientes a las cuotas de mercado y la alta actividad en los desarrollos de aplicaciones, de entre todas las plataformas más populares tanto Android como IOS parecen ser las más indicadas para realizar desarrollos de aplicaciones. Ambas están en pleno auge comiendo terreno a las demás plataformas y llevando un buen ritmo de crecimiento. A su vez al tener ambas una alta actividad de desarrollo proporciona mayores recursos y mayor movimiento en las tiendas relativo a compras y ventas.

Por lo tanto se descarta la posibilidad de desarrollar para Symbian, RIM o Windows Mobile, quedando seleccionadas Android e IOS. Antes de seleccionar una de las dos es interesante mirar y comparar sus características principales, de esa forma será más fácil tener una idea de cuál puede ser más adecuada para desarrollar la aplicación. Para ello se presenta la siguiente tabla recogiendo las características de ambas plataformas en su versión más utilizada.

Características	Android 2.2	IOS 4
<b>Características básicas</b>		
Kernel	Linux Kernel	OSX
Multitarea	Sí, multitarea completa. El SO permite correr aplicaciones en segundo plano. El Kernel controla y cierra las aplicaciones según los recursos disponibles.	Sí, pero limitado. Las aplicaciones se pausan en segundo plano. No con todas las aplicaciones.
Notificaciones Push/Cloud	Notificaciones Cloud pueden iniciar acciones en el SO sin Apps de por medio.	Notificaciones Push. Notificaciones Cloud solo con Apps “Trigger”.
Notificaciones	Sí. Con barra de múltiples notificaciones.	Sí. Sólo una notificación a la vez.
Carpetas	Sí.	Sí.
Almacenamiento Masivo	Sí.	No.
Búsqueda Integrada	Sí.	Sí.
Soporte Flash	Sí.	No.
<b>Conectividad</b>		
Tethering	Sí. Por USB y Wi-Fi.	Sí. Por Bluetooth y USB. Wi-Fi solo en móviles con Jailbreak
Voz sobre Ip	Sí. Con terceras partes.	Sí. Con terceras partes.
Hot Spot	Sí. Con Wi-Fi se puede compartir hasta con 8 dispositivos.	No.
Correo electrónico	Sí.	Sí.
Navegación GPS	Sí. Mapas gratuitos con Google Maps.	Sí. Aplicaciones de pago.
<b>Aplicaciones</b>		
Instalación de aplicaciones	Desde Market, USB o vía online.	Desde App Store o vía sincronización USB con iTunes.
Tienda de aplicaciones	Sí. Android Market.	Sí. App Store.
Actualización de aplicaciones	Individualmente, todas a la vez o automáticamente tan pronto como aparezcan las actualizaciones.	Individualmente o todas a la vez.

Tienda de música	Sí. Por terceras partes.	Sí. iTunes.
Música en Stream	Sí. Por terceras partes.	Sí. Con un plug-in del iTunes.
Juegos	Sí.	Sí.
Integración con Redes Sociales	Sí.	Sí.
Lectores de Libros	Sí. Por terceras partes.	Sí. Soporte para iBooks.
Navegador Web	Sí.	Sí.
<b>Interfaz de Usuario</b>		
Personalización de la Interfaz	Sí. Con total libertad. Permite añadir temas completos, fondos de pantalla y Widgets.	Sí. Muy reducido.
Pantalla Multitáctil	Sí.	Sí.

Tabla 1.Comparativa Android 2.2 Vs IOS 4

Sin entrar demasiado en detalles, la comparativa anterior muestra que no existe gran diferencia entre las dos plataformas. Es cierto que ambas poseen características muy similares pero sus pequeñas diferencias es lo que hace que tanto los usuarios como los desarrolladores se decidan por una de ellas.

Los usuarios valoran más las funcionalidades que los dispositivos pueden aportarles como entretenimiento, por ejemplo, aquellas englobadas en el campo multimedia como puede ser la capacidad de reproducción de vídeos y audio, y los juegos. Otras funcionalidades que se están poniendo más de moda son las que les permiten mantenerse en contacto con las redes sociales o aquellas que les facilitan la conexión y navegación por internet. Ambas plataformas cumplen bien esos requisitos, pero existe uno que las diferencia totalmente. Como usuario, la posibilidad de personalizar el interfaz del terminal es una característica a tener muy en cuenta ya que permite modelar la apariencia del terminal a gusto del usuario. En este aspecto, Android ofrece la posibilidad de una personalización completa, ya sea mediante cambios de fondo de pantalla como algo más completo que vienen siendo los temas. En cambio IOS apenas ofrece un sistema de personalización siendo éste algo pobre y mediante fondos de pantalla.

Desde el punto de vista de un desarrollador, siendo éste el que más interesa en el momento de elegir cuál de las dos plataformas se utilizará para llevar a cabo el desarrollo de la aplicación, Android es una plataforma menos restrictiva a la hora de trabajar con ella. Mientras que IOS no es código libre y se necesita pagar una licencia para poder desarrollar para ella, Android es totalmente abierto. Así mismo, otro problema de tener que trabajar con IOS implica utilizar un equipo que disponga el sistema MAC OS X teniendo por supuesto, activa la licencia, ya que en caso contrario no será posible firmar la aplicación ni podrá ser incluida en la tienda de Apple.

Otro factor a tener en cuenta como desarrollador es el lenguaje de programación de cada sistema. Android e IOS son totalmente diferentes, mientras que en Android se programa en Java, para IOS hay que saber utilizar objective-C. Aunque ambos son lenguajes de programación orientados a objetos son bastante diferentes, por lo que la programación para ambas plataformas es muy diferente.

Después de analizar todas las plataformas, comparar sus cuotas de mercado y características, se ha decidido utilizar la plataforma Android como desarrollo. Esta decisión se basa en los motivos expuestos a lo largo de esta sección. La cuota de mercado nos indica que está creciendo a un ritmo elevado, lo cual nos ofrece bastantes oportunidades. A su vez, junto con IOS tienen una alta actividad de desarrollo, lo cual proporciona un mayor número de recursos y ejemplos, y aunque sea más baja que la de IOS, esto permite que sea más fácil a la aplicación hacerse hueco en el mercado ya que hay menos competencia de aplicaciones. Ser una plataforma de código libre también ha ayudado a su elección, ya que permite trabajar con SDKs abiertos y numerosos ejemplos. Por último, que el lenguaje de programación sea Java facilita también la decisión ya que se disponen de mayores conocimientos en este lenguaje. Por todo esto, se concluye que la mejor plataforma para el desarrollo de la aplicación será Android.

# Capítulo 3

## Trabajando con Android y el calendario de Google

### 3.1 Introducción

En esta sección se encontrarán los conceptos más importantes para poder llevar a cabo el desarrollo de la aplicación, así como los problemas encontrados. Para ello es necesario tener una idea básica de cómo funciona Android, por lo que se recomienda leer la información contenida en la sección apéndice [Introducción] de este documento.

Así mismo, como se va a trabajar con el servicio de Google Calendar, es recomendable conocer el funcionamiento del protocolo de Google. Para ello se aconseja la lectura de la documentación del API de GoogleCalendar dónde se explica de manera muy completa dicho protocolo. Esta información se puede encontrar entre los recursos de Google Code. [13]

## 3.2 Preparar el entorno

Para empezar usar Android es necesario preparar un entorno de trabajo en el equipo en el que se va a trabajar. Para ello se necesitan una serie de herramientas y librerías que permitirán el desarrollo de esta plataforma.

Lo primero de todo, se necesita un entorno de desarrollo IDE. Debido a que el plug-in ADT que proporciona Android está principalmente desarrollado para el entorno IDE Eclipse[14] y proporciona una serie de herramientas que facilitan el desarrollo, hace que esta plataforma sea la más aconsejable. Una vez instalado este entorno, es necesario obtener el SDK de Android correspondiente a la plataforma sobre la que se va a trabajar, ya sea bien Windows, Linux o Mac OS X.

Si se decidió utilizar IDE Eclipse, se deberá instalar las herramientas de desarrollo de Android que se encuentran en el plug-in ADT. Para ello desde el mismo Eclipse mediante el menú **Help** y la opción **Install new software**, se descargará el plug-in.

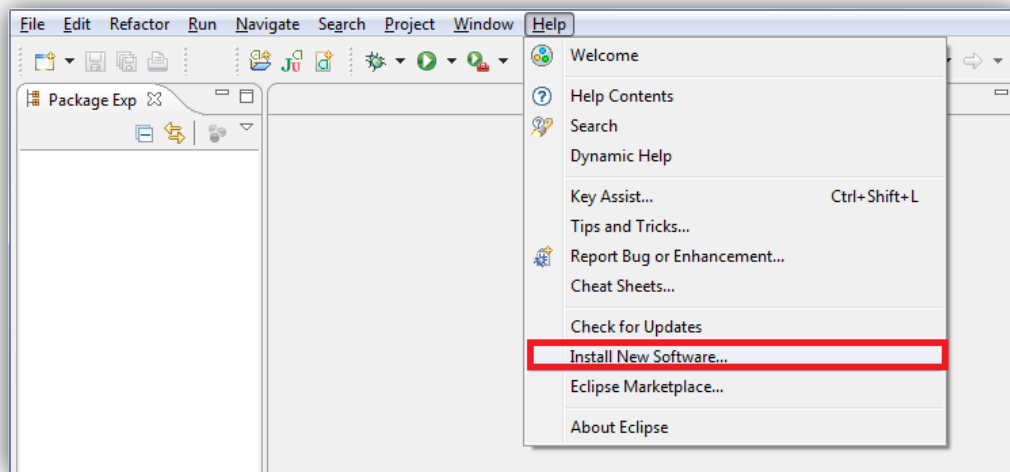


Figura 9. Instalar nuevo software en Eclipse

Se abrirá una nueva ventana y se deberá indicar la localización del plug-in. Éste se encuentra en <https://dl-ssl.google.com/android/eclipse/>, por lo que se introducirá esta url en el cuadro de texto y aparecerán una serie de elementos que deberán seleccionarse. Al continuar con el proceso será necesario aceptar las licencias de todos los elementos para acto seguido ser descargados e instalados. Por último, para finalizar la instalación y lograr que los cambios se hayan realizado de manera satisfactoria, se debe reiniciar Eclipse.

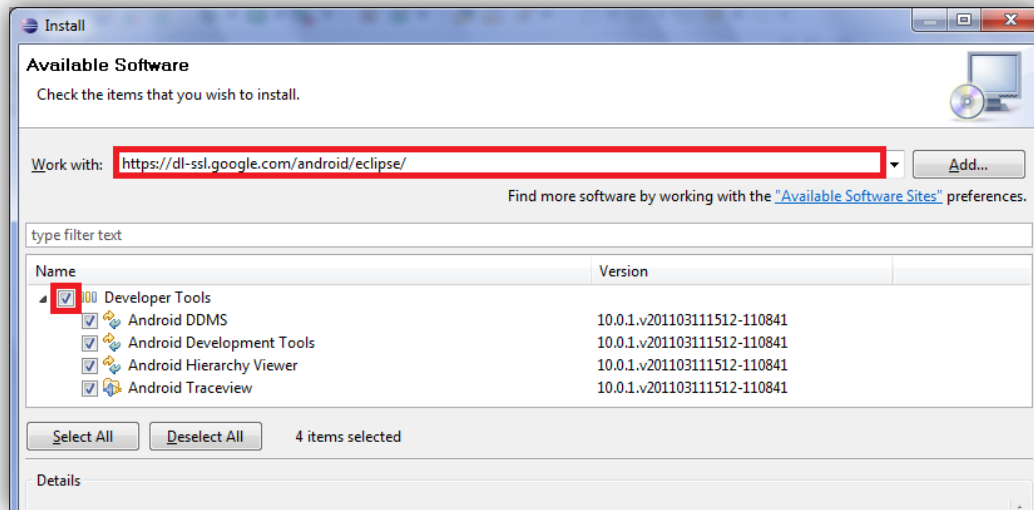


Figura 10. Descarga e instalación del plug-in

Una vez instalado el ADT y descargado y descomprimido el SDK, debe configurarse el plug-in indicándole la ubicación del SDK de Android. Para ello desde el menú **Window** se seleccionará la opción **Preferences**.

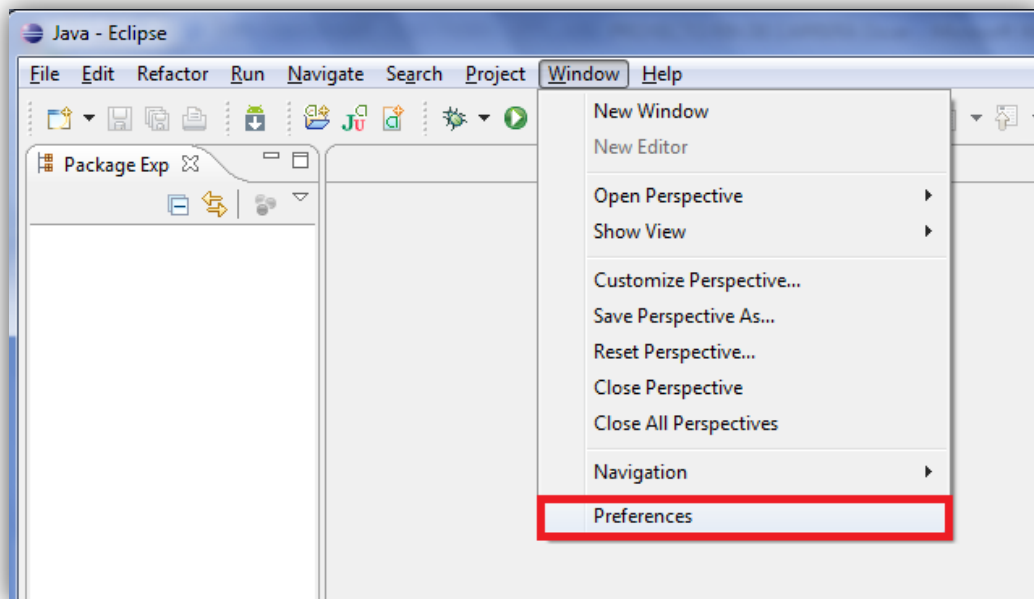


Figura 11. Configurar plug-in



En la nueva ventana, en las preferencias de Android, se indicará la ruta en la cual se ha descomprimido el SDK. Una vez configurada la ruta, se deberán instalar las actualizaciones y nuevos paquetes para asegurar un correcto funcionamiento del entorno. Para ello se abrirá *Android SDK and AVD Manager* mediante el acceso en la barra de herramientas que se ha creado tras la instalación y se seleccionarán los paquetes deseados.

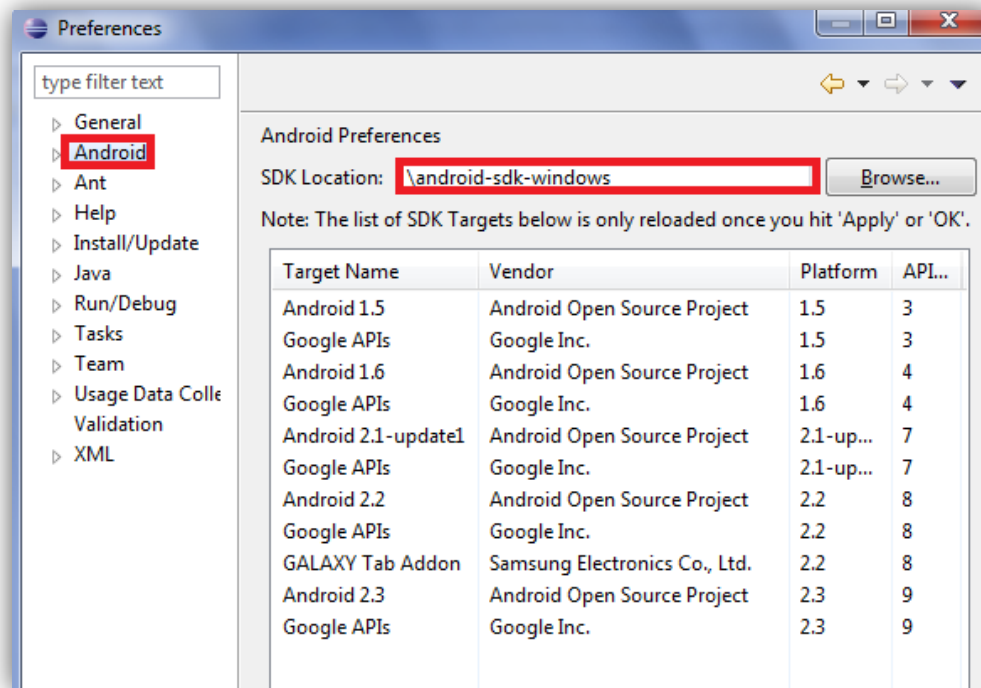


Figura 12. Referenciar SDK

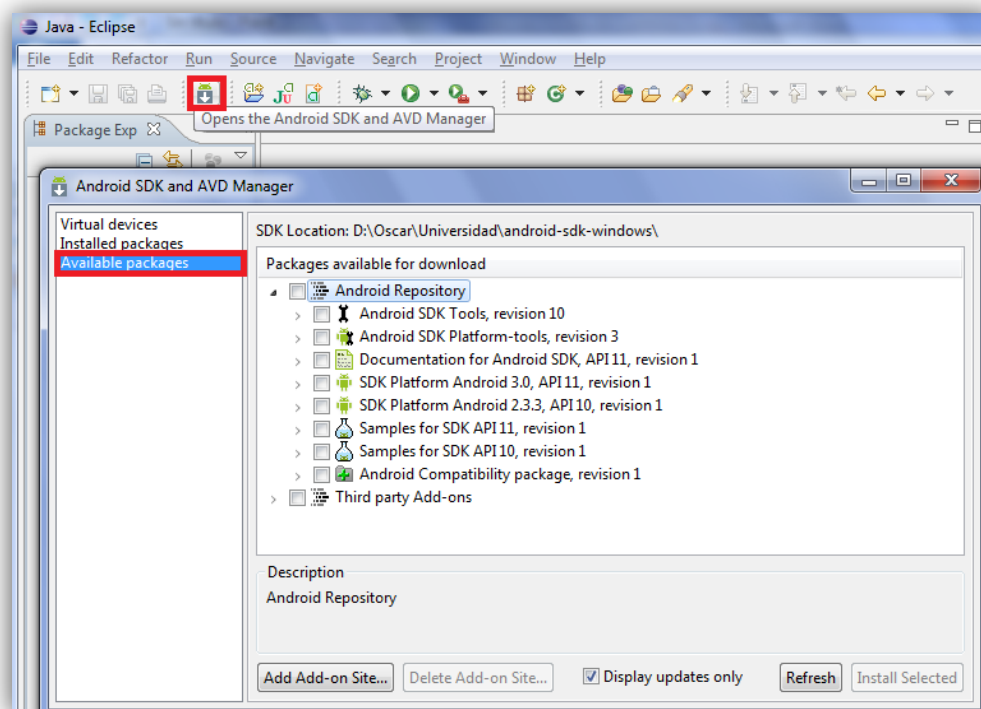


Figura 13. Descarga de nuevos paquetes

Si se han seguido correctamente los pasos, ahora se dispondrá de un entorno de trabajo para empezar a desarrollar en Android. Es cierto que también existen más componentes como códigos de ejemplo que pueden ser añadidos, o drivers que pueden facilitar la tarea del desarrollador, uno de los cuales es el driver para el USB que permitirá poder realizar pruebas en un dispositivo móvil con sistema operativo Android.

## 3.3 Uso de Librerías externas

Como ya se ha explicado en otras secciones, desarrollar para Android implica utilizar el lenguaje de programación Java, lo cual facilita la tarea al ser un lenguaje ya conocido y al disponer de numerosa documentación y librerías de fácil acceso.

Durante el desarrollo de esta aplicación se ha necesitado la disposición de dos librerías externas a Android. Por un lado, al haberse implementado una funcionalidad que permite conectarse a un servidor de correo, descargar y enviar mensajes electrónicos, es necesario el uso de la librería externa JavaMail, que proporciona un framework que permite construir mensajes y conectarse con servidores de correo. Por otro lado, al querer interactuar con el servicio del calendario de Google, es necesaria la librería que proporciona Google para Java.

### ❖ Librería JavaMail

En un primer momento se pensó en utilizar la librería que proporciona Oracle con el nombre JavaMail[16], pero resultó que al estar programando para la plataforma Android, la máquina virtual Dalvik no es compatible con Java al 100% por lo que Android no permitía su uso. Esto resultó ser un gran problema ya que esta librería permitía interactuar con servidores de correo y gracias a ella se podían leer los correos, enviar nuevos y realizar otras operaciones con ellos. Al no poder disponer de la librería, implementar la funcionalidad de la aplicación que permitía al usuario interactuar con su servidor de correo su complejidad aumentaba al no disponer de las funciones para comunicarse con el servidor. Por suerte se logró dar con tres librerías de código libre creadas para Android[17] que se habían desarrollado a partir de JavaMail y que contenían las funciones necesarias y a su vez se habían eliminado aquellas que no eran soportadas por Android. Estas librerías son mail.jar, activation.jar y additionmal.jar, conteniendo esta última nuevas dependencias de javax.awt que necesitan las otras dos librerías y rempazan las anteriores que contenía JavaMail y no eran soportadas.

Gracias a estas librerías se solventaba el problema de la librería JavaMail de Oracle y evitaba tener que crear los mensajes HTTP desde cero para poder realizar las comunicaciones con el servidor.

El único inconveniente que presentan es que no existe un API de las librerías, por lo que sus funciones no están documentadas y para algunos casos se ha tenido que mirar la documentación de JavaMail e intuir el comportamiento de la función, cosa que ha provocado algún problema que otro con el protocolo de comunicación.

#### ❖ Librería Google-API-Java-Client

Como ya se ha comentado, existen librerías para Java que no pueden ser utilizadas en Android, y es por ello por lo que Google provee de una librería especial a Android para poder desarrollar aplicaciones que interactúen con los servicios de Google. Ésta librería se llama *google-api-java-client* y se corresponde a la librería ofrecida para Java pero modificada para poder ser utilizada en Android.

Esto vuelve a presentar otros problemas como con la librería anterior, ya que esta librería no contiene las mismas funciones que la utilizada para Java y no existe una documentación propia para ella por lo que hay operaciones y funciones que no se pueden realizar, o se realizan de forma distinta a lo indicado en el protocolo de Google. Estos cambios se comentan en la sección de la Implementación de la aplicación.

Hay que indicar que esta librería está compuesta por otras pequeñas librerías que implementan distintas funciones del protocolo por lo que en caso de añadirlas todas se aumentaría el tamaño de la aplicación. A continuación se muestra una imagen en la que se pueden ver las librerías necesarias para la aplicación.

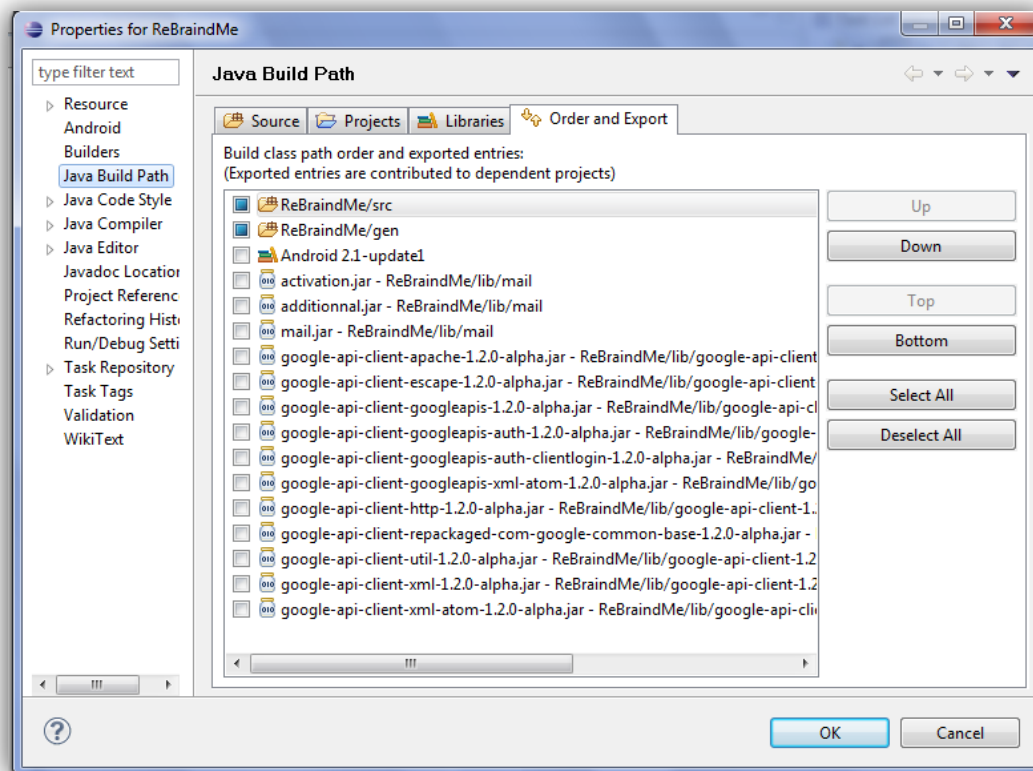


Figura 14. Librerías necesarias para GoogleCalendar

## 3.4 Interactuando con servidores de correo

Existen una serie de protocolos que permiten establecer y realizar comunicaciones con los distintos servidores de correo. Mediante ellos se permite al usuario realizar operaciones con los servidores siendo las acciones más comunes acceder a los correos, su contenido y elementos que estén adjuntos, borrar mensajes y enviar nuevos elementos.

A continuación se explicarán de forma muy breve los protocolos más utilizados, ya que se pretende comprender la funcionalidad de éstos pero sin entrar en profundidad ya que al disponerse de las librerías mencionadas, los comandos enviados se encuentran implementados en las librerías.

### ❖ Protocolo SMTP

Es un protocolo de red de nivel de aplicación utilizado para el intercambio de mensajes de correo electrónico entre dispositivos enviando el correo directamente al servidor sobre una conexión TCP/IP.

El protocolo SMTP funciona con comandos de textos enviados al puerto 25 del servidor SMTP. A cada comando enviado por el cliente le sigue una respuesta del servidor SMTP compuesta por un número y un mensaje descriptivo.

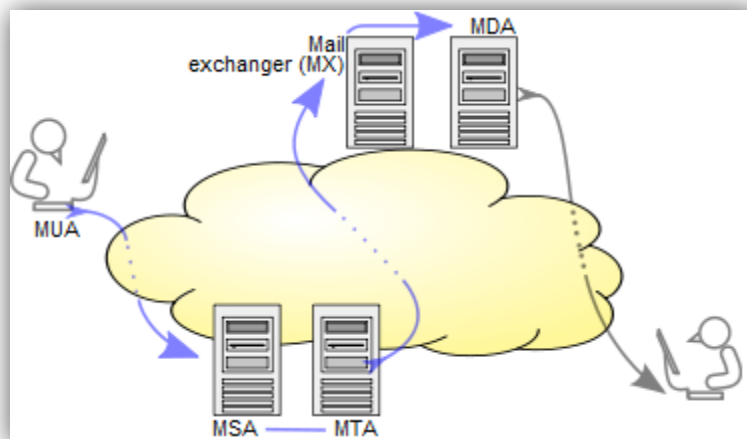


Figura 15. Envío de correo

Dentro de la aplicación Android, se utilizará para permitir al usuario enviar correos electrónicos utilizando las funciones implementadas en las librerías. En la sección de implementación se explica cómo se debe conectar con el servidor y se envía un correo.

❖ Protocolo POP3

Se trata de un protocolo de nivel de aplicación utilizado en clientes locales de correo para obtener los mensajes de correo electrónico almacenados en un servidor remoto sobre una conexión TCP/IP.

Este protocolo está diseñado para recibir correos pero no para enviarlos. Permite al cliente conectarse con el servidor y descargar los correos electrónicos eliminándolos del servidor. De esta manera es posible gestionar los correos trabajando sin conexión. Aun así, algunos clientes incluyen opciones para dejar los mensajes en el servidor.

❖ Protocolo IMAP4

Es un protocolo de red de acceso a mensajes electrónicos almacenados en un servidor. Mediante IMAP se puede tener acceso al correo electrónico desde cualquier equipo que contenga una conexión a Internet y a diferencia de POP no borra los correos del servidor ni se los descarga por lo que es menos pesado.

La siguiente tabla muestra una comparativa entre los distintos protocolos.

	<b>POP3</b>	<b>IMAP4</b>
Trabajo Offline	Sí.	No.
Trabajo Online	No.	Sí.
Número de transacciones	Baja.	Alta.
Ver correos sin descargar	No.	Sí.
Nº de carpetas gestionables	1	Ilimitado.
Buzones multiusuario (acceso múltiple)	No.	Sí.
Estados del mensaje	No.	Sí.
Búsqueda del mensaje	No.	Sí.
Gestión MIME	Básica.	Avanzada.
Mecanismo de extensión de funciones	No.	Sí.
Modos de conexión	Conectado	Conectado/Desconectado
Firma cifrada	No	Sí.
Métodos de autenticación protegida	APOP.	Nativo.

**Tabla 2. Comparativa entre POP e IMAP**

Observando la tabla puede verse que IMAP es mucho más completo que POP, incluyendo más características y funcionalidades. A pesar de ello, en un principio se intentó utilizar POP3 como protocolo de red para obtener los mensajes del servidor ya que todos los dispositivos *smartphones* incluyen este protocolo mientras que no todos incluyen IMAP. El problema fue que la implementación de las librerías equivalentes a JavaMail no funcionan correctamente con POP3, ya que daban problemas al obtener el contenido de los mensajes si éstos incluían ficheros adjuntos, aunque no se trataran, si no que simplemente se ignoraran, la lectura del contenido del mensaje provocaba errores.

Como ya se ha comentado, estas librerías no incluyen documentación ni permiten ver su contenido por lo que no podía trazarse el error. Por ello al final se ha terminado utilizando el protocolo IMAP4, el cual permite obtener los mensajes y su contenido sin problemas cuando éstos contienen datos adjuntos.

A pesar de que esto pueda reducir el mercado de la aplicación al limitar el protocolo que el dispositivo móvil deba utilizar, no es algo que preocupe demasiado ya que cada vez más dispositivos móviles incluyen IMAP4 como protocolo de transferencia junto con POP3, por lo que a lo largo del tiempo esto dejará de ser un problema.

## 3.5 Uso de los Servicios del Calendario de Google

Google proporciona una detallada documentación de su protocolo accesible a todo desarrollador que desee utilizar los servicios que ofrece para sus aplicaciones, siendo necesaria la utilización de librerías “*Java Client*” para poder acceder a las funcionalidades de dichos servicios.

Para cada lenguaje, esta librería proporciona las herramientas y capas de abstracción necesarias para poder construir las consultas y respuestas HTTP sin necesidad de crear los mensajes a mano. Cada clase de la librería proporciona las clases que se corresponden a los elementos y los tipos de datos que el API de Google utiliza. Estas herramientas se pueden descargar desde la página del Api del calendario de Google y habrá que seleccionar la apropiada dependiendo del lenguaje que se vaya a utilizar. Para esta aplicación se ha necesitado la librería para Android que como se ha comentado, se trata de una versión de Java que soporta Android pudiendo usarse para la versión 1.5 de Android en adelante.

### 3.5.1 Autenticación para servicios de Google

Una vez añadida la librería al proyecto ya se puede empezar a utilizar los servicios que ofrece el calendario de Google. Para ello, antes de obtener, añadir o eliminar cualquier dato, la aplicación, que actuará como cliente, necesita autenticarse y obtener un *token* de autenticación ya que se va a acceder a un servicio el cual está ligado a una cuenta Google que contiene información personal. Existen diferentes métodos de autenticación que se pueden utilizar para el desarrollo de una aplicación, pero al tratarse de un desarrollo en Android se analizarán únicamente los tres tipos de autenticación que se proporcionan.

#### ❖ Método de autenticación OAuth Authentication

OAuth es un protocolo abierto que permite autenticarse a través de las APIs de las aplicaciones. El objetivo de este método de autenticación es poder acceder a recursos protegidos sin la necesidad de enviar un usuario y una contraseña. Para ello, a través de la aplicación será necesario generar una firma que la identifique gracias a un conjunto de *tokens* facilitados por el servicio.

Se trata de un estándar abierto que añade una capa más de seguridad a los datos de los usuarios, pero hay que dejar claro que no pretende sustituir a otros sistemas de autenticación, si no que se integra con ellos y no puede sustituir a elementos de seguridad como los certificados SSL.

En la siguiente imagen se puede ver cómo funciona este método de autenticación y cuáles son los pasos que la aplicación debe seguir.

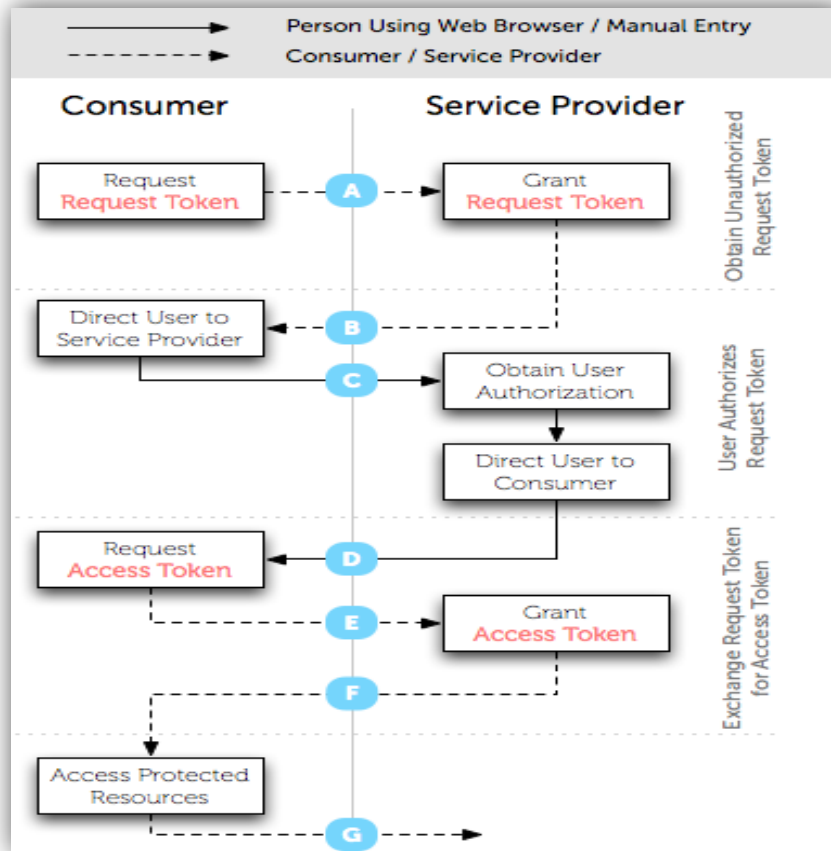


Figura 16. Autenticación OAuth2

El principal objetivo es, observando todo el proceso, conseguir un *token* de acceso (*Access Token*) y un *token* secreto (*Access Token secret*) para poder generar la firma y así poder acceder al servicio y los datos sin necesidad de enviar las credenciales de los usuarios para cada petición. Los pasos a seguir son:

1. Creación de una aplicación y obtención de una *Consumer key* y *Consumer Secret*.
2. Solicitar un *token* temporal.
3. Autorización del usuario.
4. Intercambio del *token* temporal y la autorización del usuario por un *Access Token* y *Access Secret Token*.



❖ Método de autenticación ClientLogin

No debe usarse en aplicaciones web, únicamente en aplicaciones instaladas. Es uno de los métodos más simples de autenticación y es fácil de implementar. Permite a los usuarios registrarse en sus cuentas Google y obtener acceso a sus servicios desde dentro de la aplicación.

La aplicación conecta con Google mediante los datos de autenticación del usuario, su usuario y contraseña, y solicita acceso a un servicio específico de Google. Una vez autorizada por Google, la aplicación puede acceder a los datos del servicio, permitiendo al usuario crear, leer, modificar o eliminar cualquier información que desee.

El inconveniente que posee este método de autenticación es que es necesario que el usuario introduzca sus credenciales en la aplicación, teniendo que confiar en ésta ya que manejará sus datos para poder acceder a su cuenta Google. Aun así, ofrece otras ventajas al mejorar el rendimiento ya que las credenciales no se envían cada vez que se hace una petición, si no que únicamente hace falta una validación por sesión. Gracias a esto, se reduce el número de veces que se envía esta información evitando posibles ataques. A su vez, este método también permite incorporar elementos adicionales de seguridad como puede ser el uso de *Captchas*.

En la siguiente imagen se muestra un diagrama de cómo funciona este método de autenticación.

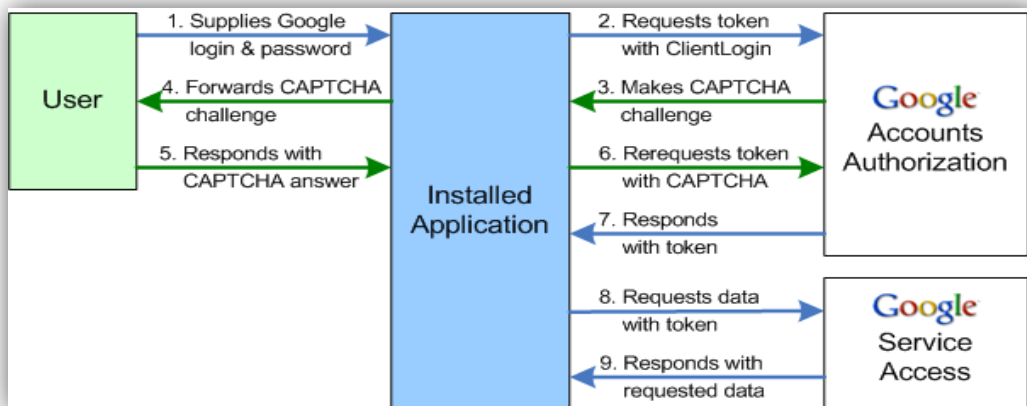


Figura 17. Autenticación ClientLogin

El primer paso de la aplicación es obtener los credenciales del usuario para acceder al servicio Google relacionado con su cuenta. Una vez adquiridos, la aplicación envía una petición ClientLogin al servicio de autorización de Google. En caso que el servicio Google requiera una verificación adicional del usuario, éste enviará un desafío *Captcha* para que la aplicación la muestre al usuario y éste la resuelva, enviando una nueva petición ClientLogin con el resultado del desafío.

Si se produce una autenticación correcta, el servicio de autorización de Google responderá a la aplicación con un *token* que será necesario para realizar las peticiones. Por lo tanto, cada vez que el servicio reconozca el *token* en cada petición, suministrará los datos a la aplicación.

#### ❖ Método de autenticación mediante Android Account Manager

En la versión del SDK de Android de 2.0 aparece por primera vez el administrador de cuentas *AccountManager* de Android. Su función es administrar todas las autenticaciones y autorizaciones en un sistema central de cuentas.

Este método de autenticación consiste en utilizar la autenticación que provee este administrador seleccionando la cuenta desde la que se desea acceder al servidor y utilizar el *token* que devuelve para poder obtener los servicios. El requisito indispensable es que debe existir dicha cuenta en el panel de control de *Cuentas y Sincronizaciones* de la configuración de Android.

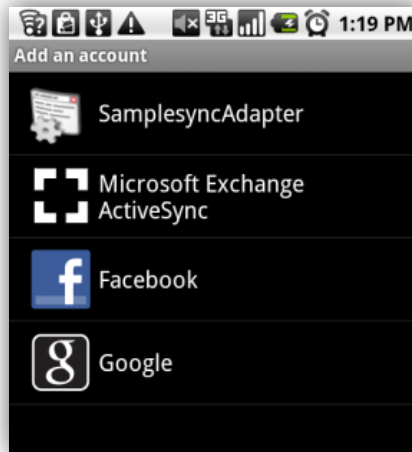


Figura 18. Panel de Control Cuentas y Sincronización

Como es de suponer, este último método de autenticación es el más adecuado y aconsejado para cualquier aplicación Android ya que deja al administrador de cuentas que tome el control en la autenticación evitando que la aplicación interactúe con las credenciales del usuario.

A pesar de todo esto, para el desarrollo de esta aplicación se ha utilizado una autenticación *ClientLogin* debido a que se han tenido problemas con el emulador proporcionado por Android al no poder acceder a este administrador por lo que no se podía implementar esta autenticación. Una solución podría haber sido sustituir el emulador por un móvil con sistema operativo Android, pero durante el desarrollo de la aplicación no se ha podido disponer de uno por lo que al final se optó por la autenticación *ClientLogin*.

Aun así, y aunque ya comentado en la sección Líneas futuras, se debe cambiar este tipo de autenticación por el *AccountManager Authentication*.

### 3.5.2 Listas de calendarios en Google Calendar

Habiendo resuelto el paso de la autenticación del usuario con el servicio de Google Calendar, ahora se debe acceder al calendario sobre el que el usuario va a trabajar. El API de Google ofrece una serie de métodos de acceso a las listas que aparecen en la aplicación web del calendario de Google. Existen tres tipos de calendarios en esas listas, un calendario principal, uno secundario y otro para calendarios importados.

El calendario principal se crea automáticamente cuando un usuario se registra en una cuenta de Google Calendar. En cuanto a los calendarios secundarios son aquellos que el usuario crea y los importados son los calendarios creados por otros usuarios a los que el usuario se suscribe.

Para obtener la lista de los calendarios del usuario basta con mandar una petición a la dirección:

```
https://www.google.com/calendar/feeds/default/allcalendars/full
```

Esa petición devolverá una lista de calendarios de entre los cuales se elegirá el calendario que se va a utilizar. Un ejemplo de estructura de calendario en el cual se puede trabajar para obtener, modificar, insertar o borrar entradas puede ser el siguiente, dónde <username> corresponde al usuario de la cuenta de Google:

```
https://www.google.com/calendar/feeds/<username>/private/full
```

Es posible la creación de nuevos calendarios para el usuario mediante mensajes creados con código Atom o JSON enviados en un mensaje POST de HTTP a la dirección *ownscalendar*

```
https://www.google.com/calendar/feeds/default/owncalendars/full
```

Siguiendo la siguiente estructura del mensaje en el que se le indica el nombre del calendario, una breve descripción y unos parámetros de configuración tanto de apariencia como formato horario.

```
<entry xmlns='http://www.w3.org/2005/Atom'
  xmlns:gd='http://schemas.google.com/g/2005'
  xmlns:gCal='http://schemas.google.com/gCal/2005'>
  <title type='text'>Little League Schedule</title>
  <summary type='text'>This calendar contains the practice schedule and game times.</summary>
  <gCal:timezone value='America/Los_Angeles'></gCal:timezone>
  <gCal:hidden value='false'></gCal:hidden>
  <gCal:color value='#2952A3'></gCal:color>
  <gd:where rel='' label='' valueString='Oakland'></gd:where>
</entry>
```

Figura 19. Código Atom para crear calendarios

### 3.5 Uso de los Servicios del Calendario de Google

También es posible la actualización de calendarios existentes mediante mensajes PUT de HTTP especificando el identificador del calendario en la dirección del comando:

```
https://www.google.com/calendar/feeds/default/owncalendars/full/  
calendarID
```

Es necesario a su vez especificar la opción de edición en el mensaje ya sea construido en Atom o en JSon-C

```
<link rel='edit' type='application/atom+xml' href='https://www.google.com/c
```

Figura 20.Sentencia de edición en Atom

```
"selfLink": "https://www.google.com/calendar/feeds/default/owncalenda:  
"canEdit": true,
```

Figura 21.Sentencia de edición en Json-C

Un ejemplo en lenguaje Atom para modificar el título y el color de un calendario es el siguiente:

```
<entry xmlns='http://www.w3.org/2005/Atom'  
  xmlns:gCal='http://schemas.google.com/gCal/2005'  
  xmlns:gd='http://schemas.google.com/g/2005'>  
  <id>http://www.google.com/calendar/feeds/default/owncalendars/full/ppgcfga9qo8jmdwan231w7s8h44  
  <published>2007-07-12T16:46:11.815Z</published>  
  <updated>2007-07-12T16:46:11.000Z</updated>  
  <title type='text'>New Title</title>  
  <summary type='text'>This calendar contains the practice schedule and game times.</summary>  
  <link rel='alternate' type='application/atom+xml' href='https://www.google.com/calendar/feeds/c  
  <link rel='http://schemas.google.com/gCal/2005#eventFeed' type='application/atom+xml' href='htt  
  <link rel='http://schemas.google.com/acl/2007#accessControlList' type='application/atom+xml' hr  
  <link rel='self' type='application/atom+xml' href='https://www.google.com/calendar/feeds/defaul  
  <link rel='edit' type='application/atom+xml' href='https://www.google.com/calendar/feeds/defaul  
  <author>  
    <name>Little League Schedule</name>  
  </author>  
  <gCal:timezone value='America/Los_Angeles' />  
  <gCal:hidden value='false' />  
  <gCal:color value='#007813' />  
  <gCal:selected value='true' />  
  <gCal:accesslevel value='owner' />  
  <gd:where valueString='Oakland' />  
</entry>
```

Figura 22.Modificar un calendario en Atom

Y por último, como es de suponer, el protocolo de Google permite eliminar los calendarios existentes mediante peticiones *DELETE* indicando la dirección *url* del calendario a eliminar.

```
DELETE /calendar/feeds/default/owncalendars/full/calendarID
```

La única al calendario primario del usuario que posee el identificador de la cuenta de correo. En caso de intentarlo se obtiene un mensaje de error de petición al intentarlo de la forma: *HTTP 400 Bad Request Error*.

#### 3.5.3 Eventos en Google Calendar

Una de las principales funcionalidades que hace que el calendario de Google sea muy utilizado por los usuarios es poder añadir elementos en fechas específicas de tal manera que permita recordar al usuario cualquier tipo de eventos que éste pueda tener.

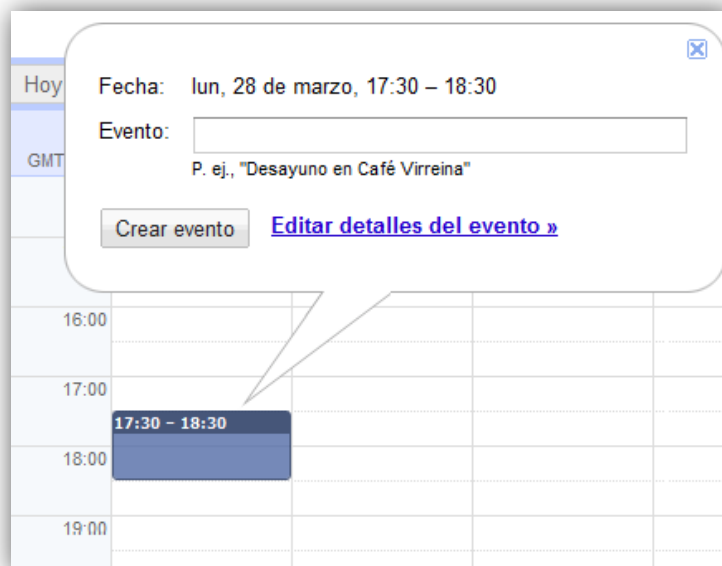


Figura 23. Crear Evento web Calendario Google

Estos eventos contienen una serie de atributos que permiten almacenar diferentes datos sobre las citas y reuniones al usuario. El lugar, una pequeña descripción e incluso la fecha de finalización son algunos de estos ejemplos.

Crear estos elementos desde la interfaz web que ofrece Google en la página del calendario es fácil e intuitivo, pero para integrar esta funcionalidad en la aplicación hay que entender cómo funcionan los eventos y cuáles son los atributos que los componen, así como sus restricciones, ya que alguno de estos atributos no son simples campos de texto.

Figura 24. Atributos de un evento

Añadir un evento a un calendario desde una aplicación externa a Google Calendar es algo más complejo ya que conlleva primero de todo, conocer y obtener el calendario en el que se va crear el evento, seguidamente se necesitará crear un mensaje que represente al evento en código Atom o JSON y por último utilizar el token de autenticación obtenido en el proceso de autenticación de la aplicación para enviar la petición de creación.

Crear un evento requiere construir un mensaje que contenga los atributos necesarios para poder añadir el evento en el calendario. Para conocer una estructura básica de cómo ha de ser un evento a continuación se muestra un fragmento de código Atom.

```
<entry xmlns='http://www.w3.org/2005/Atom'
  xmlns:gd='http://schemas.google.com/g/2005'>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/g/2005#event'></category>
  <title type='text'>Tennis with Beth</title>
  <content type='text'>Meet for a quick lesson.</content>
  <gd:transparency
    value='http://schemas.google.com/g/2005#event.opaque'>
  </gd:transparency>
  <gd:eventStatus
    value='http://schemas.google.com/g/2005#event.confirmed'>
  </gd:eventStatus>
  <gd:where valueString='Rolling Lawn Courts'></gd:where>
  <gd:when startTime='2006-04-17T15:00:00.000Z'
    endTime='2006-04-17T17:00:00.000Z'></gd:when>
</entry>
```

Figura 25. Creación de un Evento en Atom

Actualizar un evento de un calendario es muy similar a actualizar un calendario, ya que se debe obtener el evento existente, modificarlo y enviarlo de nuevo con un mensaje *PUT* a la dirección *URL* del calendario en el que se encontraba.

Es importante indicar el identificador del evento a modificar, ya que se podrían sobrescribir elementos no deseados. A continuación se muestra un ejemplo de la modificación de un evento en un calendario en el cual se indica su *ID* dentro del contenido del evento.

```
<entry xmlns='http://www.w3.org/2005/Atom'
  xmlns:gd='http://schemas.google.com/g/2005'
  xmlns:gCal='http://schemas.google.com/gCal/2005'
  gd:etag='FkkOQgZGeip7ImA6WhVR'>
  <id>http://www.google.com/calendar/feeds/jo@gmail.com/private/full/entryID</id>
  <published>2006-03-30T22:00:00.000Z</published>
  <updated>2006-03-28T05:47:31.000Z</updated>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/g/2005#event'></category>
  <title type='text'>Lunch with Darcy</title>
  <content type='text'>Change of plans - Let's discuss the new proposal.</content>
  <link rel='alternate' type='text/html'
    href='https://www.google.com/calendar/event?eid=aTJxcnNqbW9tcTJnaTE5cnMybmEwaW04bXMgk'
    title='alternate'></link>
  <link rel='self' type='application/atom+xml'
    href='https://www.google.com/calendar/feeds/jo@gmail.com/private/full/entryID' />
  <link rel='edit' type='application/atom+xml'
    href='https://www.google.com/calendar/feeds/jo@gmail.com/private/full/entryID' />
  ...
</entry>
```

Figura 26. Modificar un evento en Atom

La eliminación de eventos tampoco es muy diferente a la de las listas de los calendarios. Para ello es necesario conocer el evento a borrar, y una vez obtenida la dirección se envía un mensaje *DELETE* con la *URL* del evento.

Ya se conocen los aspectos básicos del protocolo de Google y como construir los mensajes para poder interaccionar con los eventos y calendarios del calendario de Google del usuario. Por suerte para los desarrolladores, las librerías que proporciona Google evitan, gracias a la capa de abstracción que proveen, tener que trabajar a nivel de mensajes HTTP. La forma de trabajar se explica en el apartado de 73 pero se adelanta que aunque la documentación proporcionada por Google es para programación en Java, esto permitía conocer como funcionan las librerías, pero no se podía seguir para Android ya que la versión disponible de la librería para esta plataforma no es la misma que para Java, por lo que se han tenido que realizar diversas modificaciones para poder utilizar estos servicios complicando el desarrollo al surgir nuevos problemas a los que ha habido que enfrentarse.



## 3.6 Almacenamiento de los datos

Android proporciona diferentes opciones para almacenar datos de la aplicación. La elección depende de las necesidades así como de los requisitos y del rendimiento de la aplicación pudiendo ser si la información debe ser privada, accesible por otras aplicaciones o cuánto espacio se necesita y de cuánto se dispone.

Se pueden utilizar preferencias compartidas en donde se utiliza la clase *SharedPreferences* que permite almacenar y obtener parejas de valor-clave de datos primitivos. El problema que esta opción tiene es que si se necesita almacenar datos complejos no es válida.

Otra opción es usar el almacenamiento interno del dispositivo para almacenar ficheros. Por defecto, los archivos almacenados en el dispositivo son privados para la aplicación por lo que otras aplicaciones encuentran estos recursos inaccesibles y cuando el usuario desinstala la aplicación, los archivos se borran.

También se permite el almacenamiento externo al dispositivo. Por ejemplo el uso de una tarjeta SD. Los ficheros almacenados son de lectura pública y pueden ser modificados por el usuario cuando activa el almacenamiento masivo mediante el USB para transferir archivos al ordenador. Hay que tener cuidado ya que estos archivos externos pueden desaparecer si el usuario monta un almacenamiento externo en un ordenador o si los elimina, por lo que no se ofrece ninguna seguridad sobre estos ficheros, por lo que antes de trabajar con estos archivos es necesario asegurarse que existen y son accesibles.

Si se posee un servicio web, Android permite utilizar la red para almacenar y recuperar datos mediante operaciones utilizando los paquetes .net de java y android.

Por último existe otra opción que es la que se ha utilizado y se basa en el uso de las bases de datos. Android proporciona soporte para bases de datos *SQLite* y cualquier base de datos que se cree puede ser accesible mediante el nombre por cualquier clase de la aplicación pero no entre aplicaciones. Esta opción permite utilizar consultas de *SQLite* utilizando los métodos *SQLiteDatabaseQuery()* que permiten utilizar parámetros facilitando las tareas de extracción o creación de datos. Estas consultas devuelven cursores de la clase *Cursor* que apuntan a las filas encontradas y facilitan la navegación a través de los datos. En el apartado de Implementación se encontrará más información.



# Capítulo 4

## Análisis, diseño e implementación de “ReBraindMe”

### 4.1 Introducción

Después de haber analizado los conceptos indicados a lo largo del documento incluyendo el apéndice sobre la plataforma Android, se va a pasar a comentar la fase de desarrollo del proyecto abordando su análisis diseño e implementación.

En primer lugar en el análisis se estudiarán los requisitos de usuario de la aplicación así como los diferentes casos de uso que se pueden dar. Con ello se pasará a la fase de diseño del proyecto entrando en la parte de la arquitectura del mismo. Y por último se comentarán los aspectos más relevantes de la implementación de la aplicación.

## 4.2 Análisis

En esta sección se incluirán los requisitos de usuario así como los casos de uso de los mismos.

### 4.2.1 Requisitos de Usuario

A continuación se especificarán los requisitos de usuario dividiéndolos en dos categorías, requisitos de usuario capacitivos y requisitos de usuario restrictivos. Estos requisitos se especifican a continuación siguiendo un formato tabular en el cual se incluye el siguiente contenido para cada uno de ellos:

- ❖ Identificador: Identifica de manera unívoca a un requisito. Deberá seguir la siguiente nomenclatura: RU\_CAP\_XX para los requisitos capacitivos y RU\_RES\_XX para los restrictivos, donde XX corresponde al número de requisito.
- ❖ Descripción: Especificación detallada del requisito.
- ❖ Necesidad: Establece la relevancia del requisito para el proyecto. Tomará valores entre 1 y 5 siendo 5 la necesidad más alta y 1 la más baja.
- ❖ Prioridad: Establece la prioridad del requisito dentro del proyecto. Tomará valores entre 1 y 5 siendo 5 la prioridad más alta y 1 la más baja.
- ❖ Estabilidad: Especifica la sensibilidad del requisito a ser modificado, tomando valores entre “Estable” y “No Estable”.

### 4.2.1.1 Requisitos de usuario de capacidad

A continuación se muestran los requisitos de capacidad que debe cumplir la aplicación.

<b>Identificador</b>	<b>RU_CAP_01. Conexión Correo</b>
<b>Descripción</b>	El usuario debe ser capaz de poder conectarse y acceder al su servidor de correo electrónico desde la aplicación.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 3. Requisito de Capacidad 1

<b>Identificador</b>	<b>RU_CAP_02. Listar correos nuevos</b>
<b>Descripción</b>	El usuario debe ser capaz de visualizar los correos nuevos contenidos en la bandeja de entrada del servidor de correo.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 4. Requisito de Capacidad 2

<b>Identificador</b>	<b>RU_CAP_03. Listar correos leídos</b>
<b>Descripción</b>	El usuario debe poder acceder a los correos anteriormente leídos en el caso de que desee revisar su bandeja de entrada.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 5. Requisito de Capacidad 3

<b>Identificador</b>	<b>RU_CAP_04. Eliminar Correos</b>
<b>Descripción</b>	El usuario debe tener la posibilidad de eliminar los correos electrónicos que no desee almacenar en la aplicación.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 6. Requisito de Capacidad 4

<b>Identificador</b>	<b>RU_CAP_05. Enviar Correo</b>
<b>Descripción</b>	El usuario debe tener la posibilidad de enviar nuevos correos electrónicos a diferentes destinatarios con la posibilidad de incluir un título al correo y el cuerpo con el texto que desea enviar.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 7. Requisito de Capacidad 5

<b>Identificador</b>	<b>RU_CAP_06. Responder Correos</b>
<b>Descripción</b>	El usuario debe poder responder a los correos recibidos de los usuarios ya sea bien contestando al usuario que mandó el correo o bien incluyendo también a todos los destinatarios del correo.
<b>Necesidad</b>	4
<b>Prioridad</b>	4
<b>Estabilidad</b>	Estable

Tabla 8. Requisito de Capacidad 6

<b>Identificador</b>	<b>RU_CAP_07. Conexión Calendario Google</b>
<b>Descripción</b>	El usuario debe poder conectarse y acceder a su calendario en el servicio de Google Calendar.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 9. Requisito de Capacidad 7

<b>Identificador</b>	<b>RU_CAP_08. Enviar Eventos desde Correo</b>
<b>Descripción</b>	El usuario debe poder enviar eventos a su calendario de Google Calendar desde un correo en caso que desee recordar algo del contenido de dicho e-mail.
<b>Necesidad</b>	4
<b>Prioridad</b>	4
<b>Estabilidad</b>	Estable

Tabla 10. Requisito de Capacidad 8

<b>Identificador</b>	<b>RU_CAP_9. Envío automático de Evento</b>
<b>Descripción</b>	Se enviará automáticamente un evento al calendario del usuario en Google Calendar cuando se reciba un correo con la intención de guardar un evento ya sea bien una cita o una reunión siguiendo una plantilla definida.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 11. Requisito de Capacidad 9

<b>Identificador</b>	<b>RU_CAP_10. Consultar Eventos</b>
<b>Descripción</b>	El usuario podrá consultar sobre una lista de todos los eventos almacenados del calendario los detalles de un evento en concreto.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 12. Requisito de Capacidad 10

<b>Identificador</b>	<b>RU_CAP_11. Eliminar eventos</b>
<b>Descripción</b>	El usuario tendrá la opción de poder eliminar cualquiera de los eventos del calendario, seleccionandolo de la lista de eventos que se le muestra.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 13. Requisito de Capacidad 11

<b>Identificador</b>	<b>RU_CAP_12. Enviar correo desde evento</b>
<b>Descripción</b>	El usuario tendrá la opción de poder eliminar cualquiera de los eventos del calendario, seleccionandolo de la lista de eventos que se le muestra.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 14. Requisito de Capacidad 12

<b>Identificador</b>	<b>RU_CAP_13. Listar Contactos</b>
<b>Descripción</b>	El usuario debe poder ser capaz de visualizar la lista de sus contactos del teléfono móvil.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 15. Requisito de Capacidad 13

<b>Identificador</b>	<b>RU_CAP_14. Añadir nota</b>
<b>Descripción</b>	El usuario podrá añadir una nota a un contacto seleccionándolo de la lista de sus contactos desde la aplicación.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 16. Requisito de Capacidad 14

<b>Identificador</b>	<b>RU_CAP_15. Borrar nota</b>
<b>Descripción</b>	El usuario debe ser capaz de eliminar del sistema las notas añadidas a los diferentes contactos.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 17. Requisito de Capacidad 15

<b>Identificador</b>	<b>RU_CAP_16. Visualizar notas de contactos</b>
<b>Descripción</b>	El usuario podrá seleccionar a un contacto de entre la lista de contactos y así poder ver las diferentes notas que dicho contacto posee
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 18. Requisito de Capacidad 16

<b>Identificador</b>	<b>RU_CAP_17. Configurar la aplicación</b>
<b>Descripción</b>	El usuario deberá ser capaz de configurar la aplicación estableciendo las credenciales para la cuenta de correo a la que se desea conectar así como a la cuenta Google en la que desea conectar el servicio del Calendario Google.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 19. Requisito de Capacidad 17



### 4.2.1.2 Requisitos de restricción

En las siguientes tablas se muestran los requisitos de restricción que se deben tener en cuenta a la hora del desarrollo de la aplicación.

<b>Identificador</b>	<b>RU_RES_01. Plataforma Android</b>
<b>Descripción</b>	El dispositivo en el que la aplicación correrá debe tener una versión Android 2.1 o superior.
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 20. Requisito de restricción 1

<b>Identificador</b>	<b>RU_RES_02. Permisos</b>
<b>Descripción</b>	<p>A la hora de instalar la aplicación, el usuario deberá conceder una serie de permisos a la aplicación que serán necesarios para su correcto funcionamiento en el terminal. Estos permisos serán:</p> <ul style="list-style-type: none"> <li>• Acceso a Internet.</li> <li>• Acceso de lectura a la información de los contactos del terminal.</li> </ul>
<b>Necesidad</b>	5
<b>Prioridad</b>	5
<b>Estabilidad</b>	Estable

Tabla 21. Requisito de restricción 2

<b>Identificador</b>	<b>RU_RES_03. Conectividad</b>
<b>Descripción</b>	Para poder utilizar todas las funcionalidades que ofrece la aplicación es necesario tener una conexión a internet bien por Wi-fi o por 3G.
<b>Necesidad</b>	4
<b>Prioridad</b>	4
<b>Estabilidad</b>	Estable

Tabla 22. Requisito de restricción 3

<b>Identificador</b>	<b>RU_RES_04. Protocolo IMAP</b>
<b>Descripción</b>	Debido a que la aplicación utiliza como protocolo de red, IMAP 4, es necesario para un correcto funcionamiento de la aplicación, que el dispositivo móvil en el que se va a ejecutar soporte este tipo de protocolo.
<b>Necesidad</b>	4
<b>Prioridad</b>	4
<b>Estabilidad</b>	Estable

Tabla 23. Requisito de restricción 4

<b>Identificador</b>	<b>RU_RES_05. Idioma de la Intefaz</b>
<b>Descripción</b>	La interfaz mostrará todo su contenido por defecto en inglés.
<b>Necesidad</b>	4
<b>Prioridad</b>	4
<b>Estabilidad</b>	Estable

Tabla 24. Requisito de restricción 5

### 4.2.1.3 Casos de Uso

En este apartado se incluirán los diferentes casos de uso planteados para el desarrollo de la aplicación. Para ello se utilizará un diagrama en el que se presentará al usuario como actor del caso interaccionando con las diferentes actividades.

A demás se incluirá una descripción de formato tabular en la que se describirán los pasos necesarios para los diferentes escenarios planteados conteniendo los siguientes pasos:

- ❖ Identificador: Identifica de manera unívoca a un caso de uso. Deberá seguir la nomenclatura CU\_XX, donde XX corresponde al número de caso de uso.
- ❖ Actor: Agente externo que interactúa con con el sistema participando en el caso de uso.
- ❖ Objetivo: Especificación breve de lel caso de uso.
- ❖ Pre-condición: Condiciones que deben darse para la realización del caso de uso.
- ❖ Escenario: Especificación detallada de los pasos que realiza el usuario dentro del escenario.
- ❖ Escenario Alternativo: Especificación detallada de los pasos que realiza el usuario cuando puede darse una bifurcación en la ejecución normal de la aplicación como un error.
- ❖ Post condición: Condiciones que son el resultado de la ejecución del caso de uso.

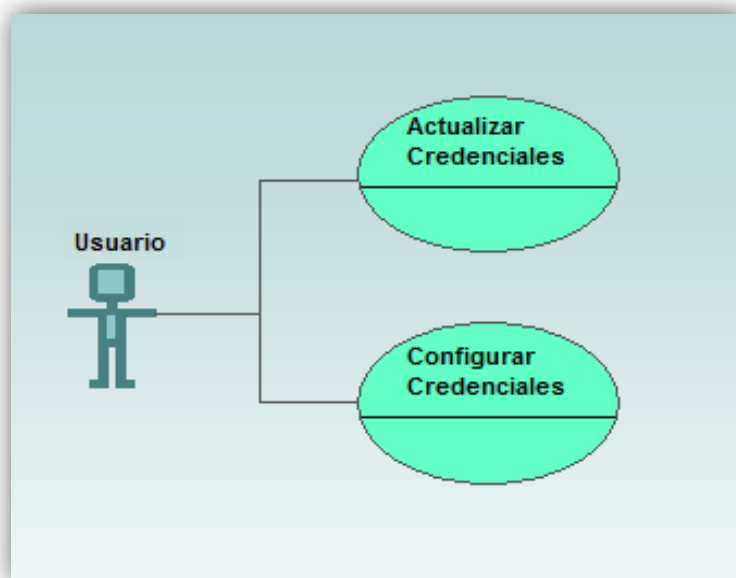


Figura 27. Caso de Uso: Configuración de la aplicación

<b>Identificador</b>	<b>CU_01. Configuración inicial de la aplicación</b>
<b>Actor</b>	Usuario de la aplicación
<b>Objetivo</b>	Configurar las diferentes credenciales para la cuenta de correo y la cuenta del Calendario de Google.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• El usuario debe haber instalado la aplicación.</li> <li>• Debe ser la primera vez que se configura la aplicación.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Desde el menú principal acceder a la configuración mediante el botón “Program Settings”.</li> <li>2. Rellenar el formulario que aparece indicando:               <ul style="list-style-type: none"> <li>• Cuenta de correo electrónico.</li> <li>• Contraseña de la cuenta de correo.</li> <li>• Cuenta del servicio del Calendario de Google.</li> <li>• Contraseña de la cuenta del Calendario de Google.</li> </ul> </li> <li>3. Salvar los datos utilizando el botón “Save Settings”.</li> </ol>
<b>Escenario Alternativo</b>	3b. Si se presiona la tecla de volver atrás no se volverá al menú principal sin salvar los datos.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• La aplicación está configurada y se puede acceder a las cuentas de correo y calendario.</li> </ul>

Tabla 25. Caso de uso CU 01.

<b>Identificador</b>	<b>CU_02. Actualización de los datos del usuario</b>
<b>Actor</b>	Usuario de la aplicación
<b>Objetivo</b>	Actualizar las credenciales de la cuenta de correo y la cuenta del Calendario Google.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>La configuración existente para las cuentas no es válida.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>Desde el menú principal acceder a la configuración mediante el botón “Program Settings”.</li> <li>Actualizar el formulario que aparece indicando: <ul style="list-style-type: none"> <li>Cuenta de correo electrónico.</li> <li>Contraseña de la cuenta de correo.</li> <li>Cuenta del servicio del Calendario de Google.</li> <li>Contraseña de la cuenta del Calendario de Google.</li> </ul> </li> <li>Salvar los datos utilizando el botón “Save Settings”.</li> </ol>
<b>Escenario Alternativo</b>	3b. Si se presiona la tecla de volver atrás no se volverá al menú principal sin salvar los datos.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>Las cuentas están actualizadas.</li> </ul>

Tabla 26. Caso de uso CU 02

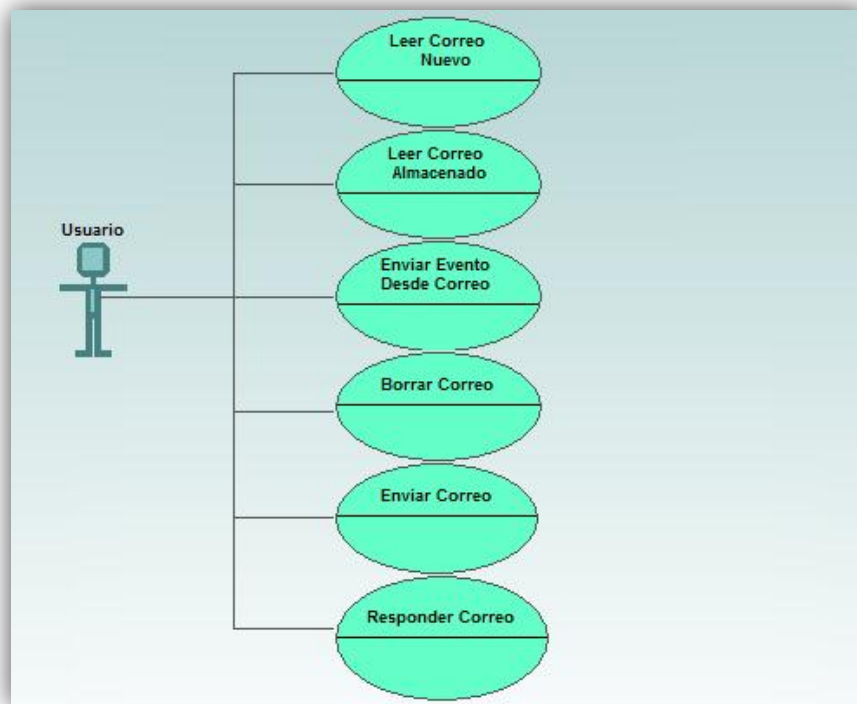


Figura 28. Caso de uso: Gestión de correo

<b>Identificador</b>	<b>CU_03. Leer correo nuevo</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Leer nuevos correos electrónicos del servidor.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe estar configurada la cuenta de correo.</li> <li>• Debe existir conexión a internet.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Desde el menú principal acceder a la gestión del correo mediante el botón “Mail Management”.</li> <li>2. Aparecerá un mensaje cuando se termine de leer y descargar el correo nuevo del servidor, que habrá que aceptar.</li> <li>3. Acceder a la bandeja de entrada de nuevos correos a través del botón “Unread Mail”.</li> <li>4. Seleccionar el correo a leer y se abrirá en una nueva ventana.</li> </ol>
<b>Escenario Alternativo</b>	<ol style="list-style-type: none"> <li>2b. Si no existe conexión a internet aparecerá un mensaje de aviso. El usuario podrá ver los correos no leídos, pero sólo aquellos que ya se habían descargado previamente.</li> </ol>
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• El mensaje leído desaparecerá del buzón de nuevos correos para mostrarse posteriormente en el buzón de correos almacenados.</li> </ul>

Tabla 27. Caso de uso CU 03

<b>Identificador</b>	<b>CU_04. Leer correo almacenado</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Leer correo electrónico almacenado.
<b>Pre-Condición</b>	
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Desde el menú principal acceder a la gestión del correo mediante el botón “Mail Management”.</li> <li>2. Aparecerá un mensaje cuando se termine de leer y descargar el correo nuevo del servidor, que habrá que aceptar.</li> <li>3. Acceder a la bandeja de entrada de nuevos correos a través del botón “Inbox”.</li> <li>4. Seleccionar el correo a leer y se abrirá en una nueva ventana.</li> </ol>
<b>Escenario Alternativo</b>	2b. Si no existe conexión a internet aparecerá un mensaje de aviso.
<b>Post-Condición</b>	Se mostrará en una ventana el correo a leer.

Tabla 28. Caso de uso CU 04

<b>Identificador</b>	<b>CU_05. Enviar evento desde un correo</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Enviar un evento desde un correo.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir conexión a internet.</li> <li>• El correo debe seguir el formato de la plantilla.</li> <li>• Debe estar configurada la cuenta del Calendario de Google.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Acceder al correo que se desea enviar como evento.</li> <li>2. Enviar mediante el botón “Add to Calendar” en el caso que no se hubiera enviado ya al calendario o “Add Again to Calendar” si el mensaje ya ha sido enviado como evento.</li> </ol>
<b>Escenario Alternativo</b>	No aplica.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• Se envía un evento al calendario del usuario.</li> </ul>

Tabla 29. Caso de uso CU 05

<b>Identificador</b>	<b>CU_06. Borrar un correo</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Eliminar un correo desde un buzón de entrada
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir el correo.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Acceder al correo que se desea eliminar.</li> <li>2. Borrar el correo mediante el botón “Delete” dentro del correo electrónico.</li> </ol>
<b>Escenario Alternativo</b>	No aplica.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• El correo se elimina del buzón y del sistema.</li> </ul>

Tabla 30. Caso de uso CU 06

<b>Identificador</b>	<b>CU_07. Enviar un correo electrónico</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Enviar un correo electrónico.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir conexión a internet.</li> <li>• Debe estar configurada la cuenta de correo del usuario.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Dentro del menú de la gestión de correo acceder a la interfaz para enviar un correo electrónico a través del botón “Send Mail”.</li> <li>2. Rellenar el formulario del correo.</li> <li>3. Enviar el correo mediante el botón “Send”.</li> </ol>
<b>Escenario Alternativo</b>	<p>3b. Si las direcciones de correo destinatarias no están mal formadas al no seguir el formato “ejemplo@ejemplo.com” dará un error y el mensaje no se enviará.</p> <p>3c. Si en lugar de enviar el mensaje se quiere descartar, se pulsará “Discard” y el correo no se enviará.</p>
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• Se envía un correo electrónico a los destinatarios.</li> </ul>

Tabla 31. Caso de uso CU 07



<b>Identificador</b>	<b>CU_08. Responder un correo electrónico</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Responder a un correo electrónico recibido.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir conexión a internet.</li> <li>• Debe estar configurada la cuenta de correo del usuario.</li> <li>• Debe existir el correo a responder.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Seleccionar el correo electrónico al que se desea responder y en la interfaz del correo electrónico interactuar con el botón “Reply”.</li> <li>2. Rellenar el formulario del correo a responder, teniendo en cuenta que se agregará de manera automática el destinatario con el valor del remitente del correo al que se está contestando.</li> <li>3. Enviar el correo mediante el botón “Send”.</li> </ol>
<b>Escenario Alternativo</b>	<ol style="list-style-type: none"> <li>1b. Si en lugar de seleccionar el botón “Reply” se utiliza el botón “Reply All” se contestará al correo agregando a todos los participantes del correo anterior.</li> <li>3b. Si las direcciones de correo destinatarias no están mal formadas al no seguir el formato “ejemplo@ejemplo.com” dará un error y el mensaje no se enviará.</li> <li>3c. Si en lugar de enviar el mensaje se quiere descartar, se pulsará “Discard” y el correo no se enviará.</li> </ol>
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• Se envía un correo electrónico a los destinatarios.</li> </ul>

Tabla 32. Caso de uso CU 08

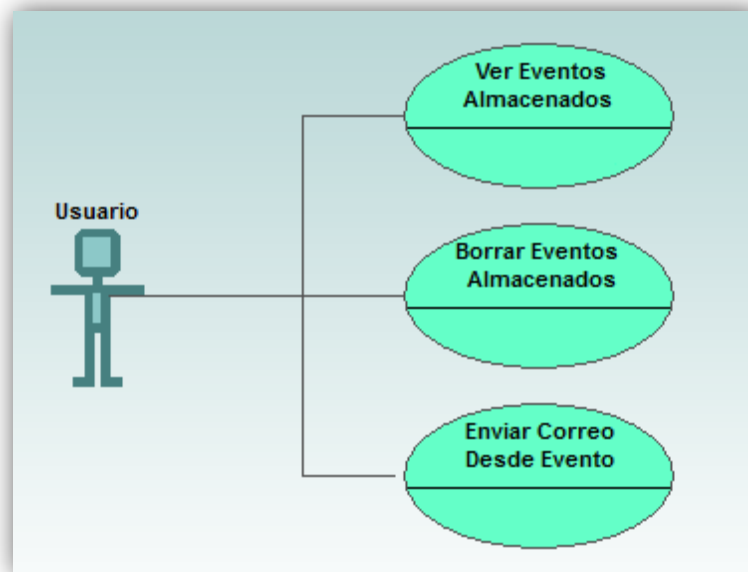


Figura 29. Caso de uso: Gestión de eventos

<b>Identificador</b>	<b>CU_09. Ver eventos almacenados.</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Acceder a los eventos almacenados enviados al calendario.
<b>Pre-Condición</b>	
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Desde el menú principal acceder a los eventos desde el botón “Remember Event”.</li> <li>2. Aparecerán los eventos en una lista mostrando el título y la hora en caso de que existan.</li> <li>3. Seleccionar el evento pulsando sobre éste y se mostrará en una ventana nueva.</li> </ol>
<b>Escenario Alternativo</b>	No aplica.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• Se mostrará en una ventana la información del evento.</li> </ul>

Tabla 33. Caso de uso CU 09

<b>Identificador</b>	<b>CU_10. Borrar un evento</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Eliminar un evento de la lista de eventos.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir el evento.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Seleccionar de la lista el evento que se desea eliminar.</li> <li>2. Se mostrará su contenido en una ventana nueva.</li> <li>3. Borrar el evento mediante el botón “Delete”.</li> </ol>
<b>Escenario Alternativo</b>	No aplica.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• El evento quedará borrado.</li> </ul>

Tabla 34. Caso de uso CU 10

<b>Identificador</b>	<b>CU_11. Enviar correo desde un evento</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Enviar un correo electrónico con la información del evento
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir conexión a internet.</li> <li>• Debe estar configurada la cuenta de correo.</li> <li>• Debe existir el evento.</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Acceder al evento al que se desea enviar como correo electrónico.</li> <li>2. Dentro de la ventana del evento pulsar el botón “Send Mail”.</li> <li>3. Rellenar los campos del formulario del correo teniendo en cuenta que el cuerpo de éste se rellenará automáticamente con los datos del evento.</li> <li>4. Enviar mediante el botón “Send”.</li> </ol>
<b>Escenario Alternativo</b>	<p>3b. Si las direcciones de correo destinatarias no están mal formadas al no seguir el formato “ejemplo@ejemplo.com” dará un error y el mensaje no se enviará.</p> <p>3c. Si en lugar de enviar el mensaje se quiere descartar, se pulsará</p>

	“Discard” y el correo no se enviará.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>Se envía un correo electrónico con la información del evento a los destinatarios.</li> </ul>

Tabla 35. Caso de uso CU 11

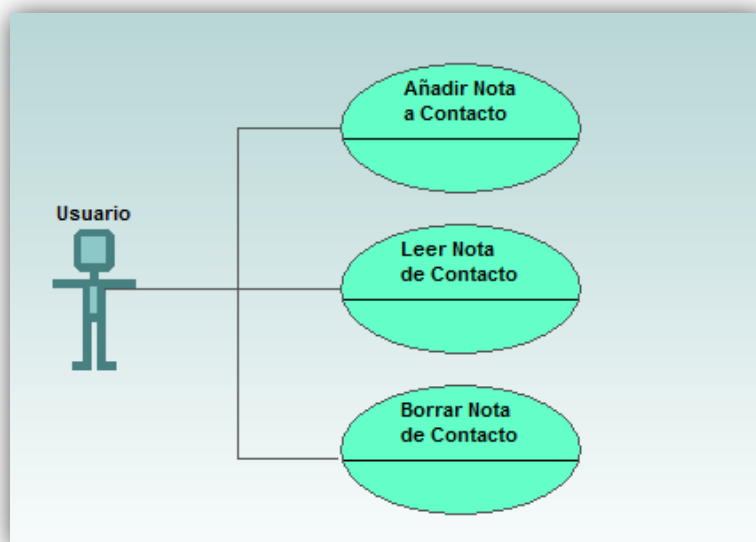


Figura 30. Caso de uso: Gestión de notas

<b>Identificador</b>	<b>CU_12. Añadir nota a un contacto</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Añadir una nota a un contacto del usuario del teléfono móvil
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>Debe existir el contacto en la agenda del usuario</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>Desde el menú principal acceder a los contactos desde el botón “Contact Notes”.</li> <li>De la lista de contactos, seleccionar aquel al que se desea añadir una nota.</li> <li>Escribir la información que se desea almacenar para el contacto.</li> <li>Guardar la nota mediante el botón “Save Note”.</li> </ol>
<b>Escenario Alternativo</b>	4b. Si en lugar de guardar la nota se quiere descartar, se pulsará “Discard” y la nota no se almacenará
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>Se almacena una nota para ese contacto</li> </ul>

Tabla 36. Caso de uso CU 12

<b>Identificador</b>	<b>CU_13. Leer nota de un contacto</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Acceder y leer las notas almacenadas de un contacto.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir el contacto en la agenda del usuario</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Desde el menú principal acceder a los contactos desde el botón “Remember Contact”.</li> <li>2. De la lista de contactos, seleccionar sobre el que se desea consultar.</li> <li>3. Aparecerán en una lista las notas que se han almacenado del contacto.</li> </ol>
<b>Escenario Alternativo</b>	No aplica.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• Se muestran en una lista las notas con el cuerpo de la nota.</li> </ul>

Tabla 37. Caso de uso CU 13

<b>Identificador</b>	<b>CU_14. Borrar una nota</b>
<b>Actor</b>	Usuario de la aplicación.
<b>Objetivo</b>	Eliminar una nota almacenada para un contacto.
<b>Pre-Condición</b>	<ul style="list-style-type: none"> <li>• Debe existir el contacto en la agenda del usuario</li> </ul>
<b>Escenario</b>	<ol style="list-style-type: none"> <li>1. Desde el menú principal acceder a los contactos desde el botón “Remember Contact”.</li> <li>2. De la lista de contactos, seleccionar sobre el que se desea consultar.</li> <li>3. Aparecerán en una lista las notas que se han almacenado del contacto.</li> <li>4. Mantener pulsado sobre la nota que se desea eliminar para que aparezca el menú contextual “Delete Note”.</li> <li>5. Pulsar sobre éste para eliminar la nota.</li> </ol>
<b>Escenario Alternativo</b>	No aplica.
<b>Post-Condición</b>	<ul style="list-style-type: none"> <li>• Se eliminará la nota del sistema.</li> </ul>

Tabla 38. Caso de uso CU 14

## 4.3 Diseño

Después de haber realizado un análisis de los requisitos y establecer las funcionalidades de la aplicación, se debe definir un diseño de la misma dentro del ciclo de desarrollo de software. Para ello se describirá de manera detallada un diseño de la arquitectura de la aplicación, un diseño de la base de datos y se comentarán los diferentes diseños realizados para la interfaz de usuario.

### 4.3.1 Diseño de la Arquitectura

En este apartado se analizará la arquitectura que debe tomar la aplicación así como un estudio de los componentes que la fomarán. Esto permitirá trazar los requisitos descritos en la fase de análisis de tal manera que se pueda verificar que la implementación realizada cumple con todas las funcionalidades previamente descritas.

La aplicación utilizará una arquitectura de diseño basada en Modelo-Vista-Controlador (MVC), de manera que se consigue separar los datos de la interfaz de usuario y de la lógica de control de la misma. Esto permitirá una mayor transparencia y depuración así como una mayor reutilización de componentes y facilidad de desarrollo.

Para diseñar un diagrama de componentes y establecer los diferentes elementos que aparecerán en él, es necesario realizar una división a alto nivel de las aplicación. Ésta puede ser dividida en tres módulos, cada uno correspondiente a las principales funcionalidades que deberá llevar a cabo. Por un lado deberá intecatuar con el servidor de correo, por otro lado deberá acceder a los servicios de Google y por último deberá ser capaz de utilizar los contactos del dispositivo móvil para insertar notas a los diferentes contactos. La siguiente figura ilustra de manera gráfica esta división de módulos.



Figura 31. Componentes de la aplicación

La siguiente figura se corresponde con el diagrama de componentes de la aplicación. En él se representan los principales componentes del sistema así como las relaciones entre ellos. Se puede observar la separación de capas de la arquitectura MVC, teniendo por un lado la parte del modelo de datos correspondiendo con el paquete “*Data Storage*”, la vista de la aplicación mediante el componente “*UI*” y por último el controlador de la aplicación en el cual se encuentran el resto de los diferentes componentes.

Será esta parte en la que se centrará el diseño de arquitectura para estudiar los diferentes elementos que lo componen.

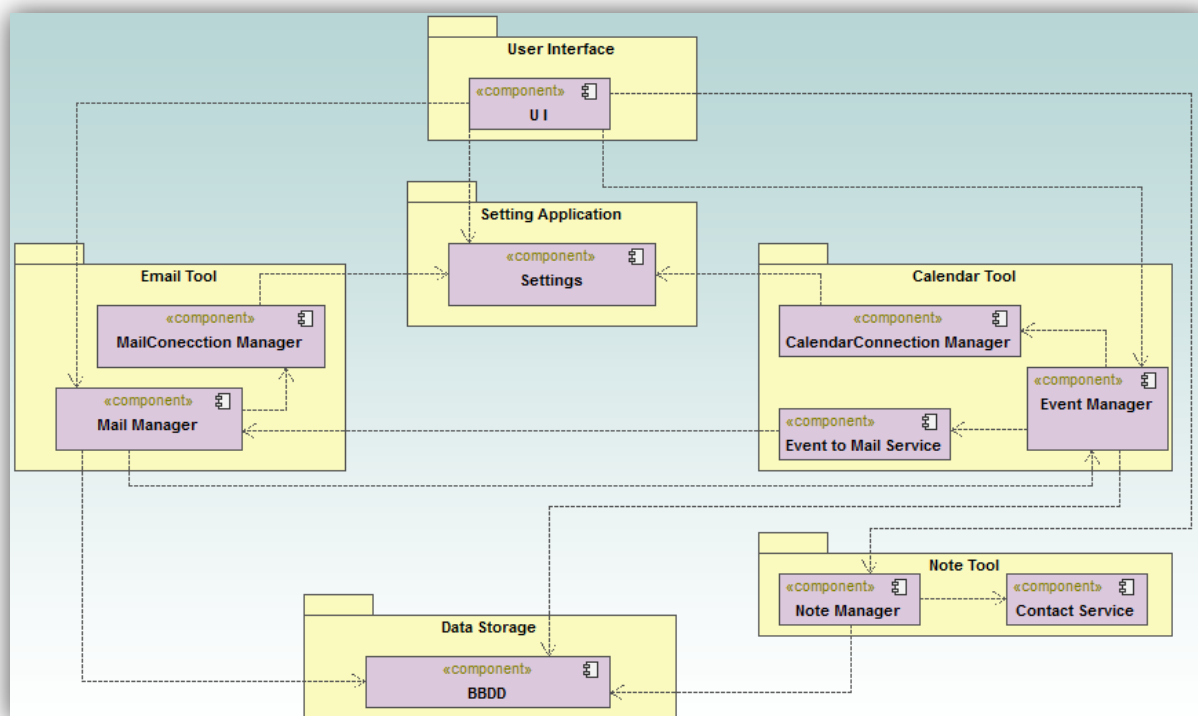


Figura 32. Diagrama de Componentes

Para cada uno de estos elementos se incluirá una especificación de los mismos siguiendo un formato tabular que contendrá la siguiente información:

- ❖ Identificador: Identifica de manera unívoca a un componente. Deberá seguir la nomenclatura COM\_XX, donde XX corresponde al número del componente.
- ❖ Descripción: Breve descripción del componente.
- ❖ Acciones: Listado de las acciones que realiza el componente.
- ❖ Dependencia: Listado de los componentes de los que depende.
- ❖ Requisitos: Relación de los requisitos que generan este componente.



<b>Identificador</b>	<b>COM_01. MailConnector Manager</b>
<b>Descripción</b>	Este componetne se encarga de administrar la conexión con el servidor de correo electrónico. Debe permitir conectar al usuario con su cuenta de correo y poder obtener los mensajes almacenados en las carpetas del servidor.
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Obtener credenciales usuario</li> <li>• Realizar conexión con el Servidor</li> </ul>
<b>Dependencia</b>	<b>COM_08</b>
<b>Requisitos</b>	RU_CAP_01, RU_RES_01, RU_RES_02, RU_RES_03, RU_RES_04

Tabla 39. Componente MailConnector Manager

<b>Identificador</b>	<b>COM_02. Mail Manager</b>
<b>Descripción</b>	<p>Este componetne se encarga de administrar las acciones sobre los correos tales como recuperarlos del servidor, enviar nuevos correos o almacenarlos en el dispositivo entre otros. Para ello es necesario que previamente se haya establecido una conexión con el servidor.</p> <p>A su vez también deberá analizar los mensajes de correo para comprobar si siguen la estructura definida de un evento, y en caso de ser así, deberá utilizar las funcionalidades proporcionadas por el componente “COM_04” para poder enviar un nuevo evento al calendario.</p>
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Abrir Bandeja de Entrada</li> <li>• Leer y obtener los emails en la bandeja de entrada</li> <li>• Almacenar los correos en el dispositivo</li> <li>• Visualizar correos electrónicos nuevos y leídos</li> <li>• Borrar emails</li> <li>• Cambiar estado de mensajes de no leído a leído</li> <li>• Enviar nuevo correo electrónico</li> <li>• Refrescar buzón de entrada de la aplicación</li> <li>• Analizar estructura del email</li> <li>• Enviar nuevo evento al Calendario de Google</li> </ul>
<b>Dependencia</b>	<b>COM_01, COM_04.</b>
<b>Requisitos</b>	RU_CAP_01, RU_CAP_02, RU_CAP_03, RU_CAP_04, RU_CAP_05, RU_CAP_06, RU_CAP_08, RU_CAP_09, RU_RES_01, RU_RES_02, RU_RES_03, RU_RES_04, RU_RES_05

Tabla 40. Componente Mail Manager

<b>Identificador</b>	<b>COM_03. CalendarConnector Manager</b>
<b>Descripción</b>	Este componetne se encarga de administrar la conexión con el servidor de google para poder acceder al servicio del calendario. Debe permitir conectar al usuario con su cuenta asignada al servicio Google y así poder interactuar con los calendarios que posee.
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Obtener credenciales usuario</li> <li>• Obtener Token de Autenticación</li> <li>• Realizar conexión con el servicio</li> </ul>
<b>Dependencia</b>	<b>COM_08</b>
<b>Requisitos</b>	RU_CAP_07, RU_RES_01, RU_RES_02, RU_RES_03

Tabla 41. Componente CalendarConnector Manager

<b>Identificador</b>	<b>COM_04. Event Manager</b>
<b>Descripción</b>	<p>Este componetne se encarga de administrar las acciones sobre los eventos dentro de un calendario del usuario en el servicio Google.</p> <p>A su vez podrá ofrecer al usuario la posibilidad de componer un correo electrónico a partir de un evento.</p>
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Acceder al calendario</li> <li>• Crear un evento</li> <li>• Enviar un evento</li> <li>• Almacenar un evento en el dispositivo</li> <li>• Visualizar un evento</li> <li>• Borrar un evento</li> <li>• Acceso a nuevo correo electrónico</li> </ul>
<b>Dependencia</b>	<b>COM_3, COM_05.</b>
<b>Requisitos</b>	RU_CAP_07, RU_CAP_09, RU_CAP_10, RU_CAP_11, RU_CAP_12, RU_RES_01, RU_RES_02, RU_RES_03, RU_RES_05

Tabla 42. Componente Event Manager

<b>Identificador</b>	<b>COM_05. Event Mail Service</b>
<b>Descripción</b>	Este componente servirá como intermediario entre el componente COM_4 y el componente COM_2, generando un correo electrónico a partir de un evento y permitiendo al usuario personalizarlo, de tal manera que se permita compartir eventos entre los contactos de correo del usuario de la aplicación.
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Generar correo electrónico</li> <li>• Enviar email</li> </ul>
<b>Dependencia</b>	<b>COM_2.</b>
<b>Requisitos</b>	RU_CAP_12, RU_RES_01, RU_RES_02, RU_RES_03, RU_RES_05

Tabla 43. Componente Event Mail Service

<b>Identificador</b>	<b>COM_06. Contact Service</b>
<b>Descripción</b>	Se trata de un componente que deberá obtener los contactos almacenados en la agenda del dispositivo para ser seleccionados en caso que se desee añadir algún recordatorio en forma de nota.
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Obtener contactos</li> <li>• Listar contactos</li> </ul>
<b>Dependencia</b>	
<b>Requisitos</b>	RU_CAP_13, RU_RES_01, RU_RES_05

Tabla 44. Componente Contact Service

<b>Identificador</b>	<b>COM_07. Note Manager</b>
<b>Descripción</b>	Componente que permitirá seleccionar un contacto de la lista y añadir o borrar notas de recordatorio.
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Crear nota</li> <li>• Eliminar Nota</li> <li>• Visualizar Nota</li> </ul>
<b>Dependencia</b>	<b>COM_06</b>
<b>Requisitos</b>	RU_CAP_14, RU_CAP_15, RU_CAP_16, RU_RES_01, RU_RES_05

Tabla 45. Componente Note Manager

<b>Identificador</b>	<b>COM_08. Settings</b>
<b>Descripción</b>	Este componente se encarga de la configuración de la aplicación. Se gestionarán las configuraciones de las cuentas del usuario, tanto de correo como del servicio google.
<b>Acciones</b>	<ul style="list-style-type: none"> <li>• Insertar usuario y contraseña para el servidor de correo</li> <li>• Insertar usuario y contraseña para el servicio Google</li> <li>• Actualizar credenciales</li> </ul>
<b>Dependencia</b>	No Aplica
<b>Requisitos</b>	RU_CAP_17, RU_RES_01, RU_RES_05

Tabla 46. Componente Settings

Una vez definidos los componentes, con el fin de evitar que se hayan pasado por alto los requisitos tomados durante la fase de análisis, a continuación se muestra una matriz de trazabilidad entre requisitos y componentes.

<div>Componente</div> <div>Requisito</div>	COMP_01	COMP_02	COMP_03	COMP_04	COMP_05	COMP_06	COMP_07	COMP_08
RU_CAP_01	X	X						
RU_CAP_02		X						
RU_CAP_03		X						
RU_CAP_04		X						
RU_CAP_05		X						
RU_CAP_06		X						
RU_CAP_07			X	X				

<b>RU_CAP_08</b>		<b>X</b>						
<b>RU_CAP_09</b>		<b>X</b>		<b>X</b>				
<b>RU_CAP_10</b>				<b>X</b>				
<b>RU_CAP_11</b>				<b>X</b>				
<b>RU_CAP_12</b>				<b>X</b>	<b>X</b>			
<b>RU_CAP_13</b>						<b>X</b>		
<b>RU_CAP_14</b>							<b>X</b>	
<b>RU_CAP_15</b>							<b>X</b>	
<b>RU_CAP_16</b>							<b>X</b>	
<b>RU_CAP_17</b>								<b>X</b>
<b>RU_RES_01</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>
<b>RU_RES_02</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>			
<b>RU_RES_03</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>			
<b>RU_RES_04</b>	<b>X</b>	<b>X</b>						
<b>RU_RES_05</b>		<b>X</b>		<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>	<b>X</b>
<div> <div>Componente</div> <div>Requisito</div> </div>	<b>COMP_01</b>	<b>COMP_02</b>	<b>COMP_03</b>	<b>COMP_04</b>	<b>COMP_05</b>	<b>COMP_06</b>	<b>COMP_07</b>	<b>COMP_08</b>

Tabla 47. Matriz Trazabilidad Requisitos-Componentes

### 4.3.2 Diseño de la Base de Datos

Debido a los requisitos obtenidos durante la fase de análisis, es necesario el uso de un sistema de almacenamiento para poder manejar toda la información con la que la aplicación va a trabajar. A pesar de que únicamente se almacenará texto, se deberán guardar los datos correspondientes a los correos, eventos y notas entre otros, haciendo que la aplicación vaya consumiendo más espacio con el paso del tiempo. Otra característica muy importante de la información almacenada es que debe ser fácilmente accesible para poder trabajar con ella, por lo que se ha optado por el uso de una base de datos.

Para ello se ha decidido utilizar las funcionalidades que ofrecen Android y SQLite que permite acceder a los datos mediante consultas de manera rápida y sencilla. Esta base de datos debe contener las siguientes tablas con sus correspondientes atributos:

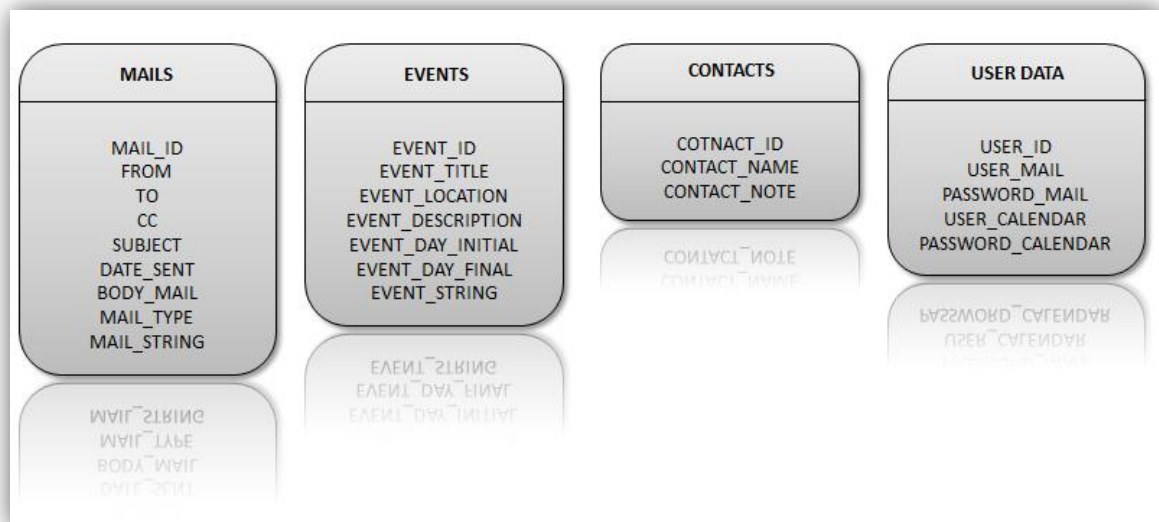


Figura 33. Tablas Base de Datos

### 4.3.3 Diseño de la Interfaz

La intención de esta sección es mostrar las diferentes interfaces desarrolladas para la aplicación y a su vez comentar su funcionamiento de tal manera que sirva como pequeño manual de usuario para la aplicación.

Para el inicio de la aplicación, se muestra una interfaz inicial con el logo de la aplicación de fondo así como un cargando para evitar que el usuario sufra una espera al iniciarse sin saber que está ocurriendo. Seguidamente se mostrará el menú principal permitiendo al usuario seleccionar la funcionalidad a la que desea ir dentro de la aplicación.

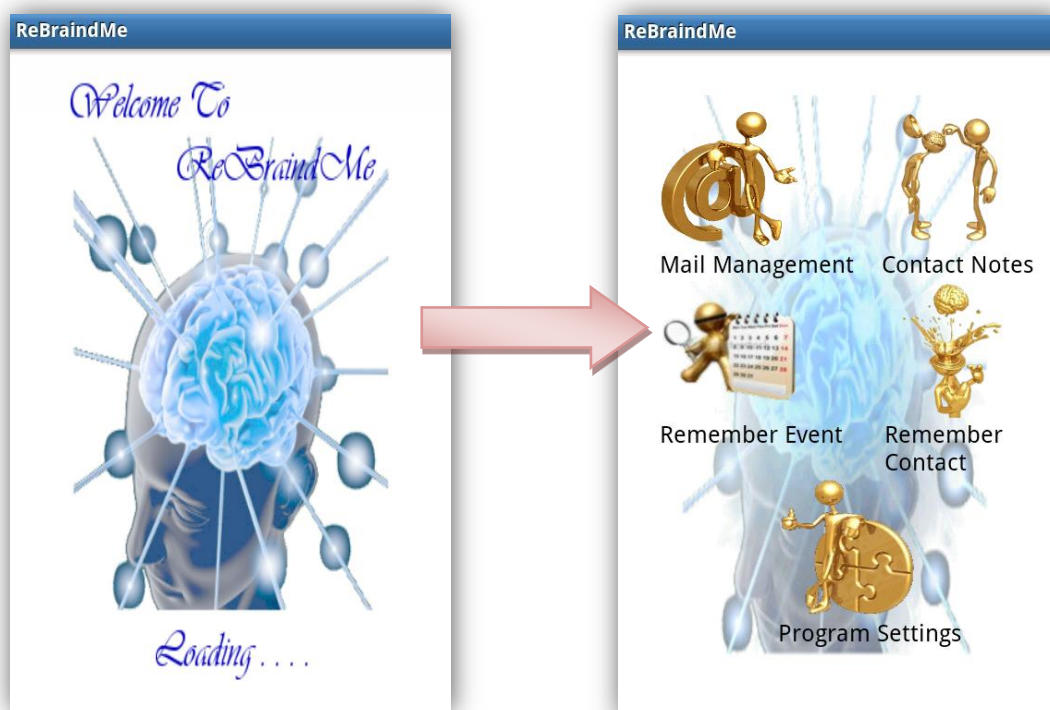
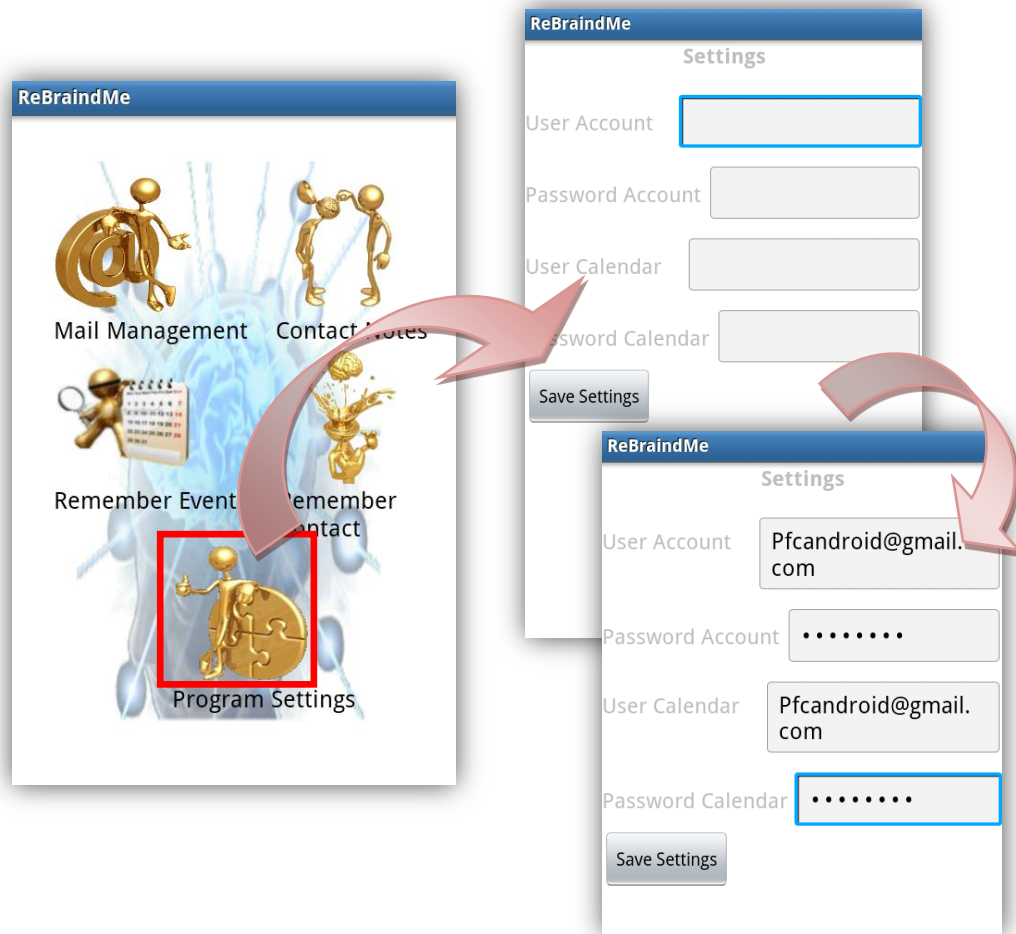


Figura 34. Inicio Aplicación Menú Principal

Como se puede observar en la ventana del menú principal, el usuario tiene cinco opciones de entre las que puede elegir. Nada más instalar la aplicación, la primera vez que el usuario inicia la aplicación es recomendable que se dirija a la configuración de la misma, ya que aunque la aplicación está preparada para utilizarse sin configurar los parámetros de las cuentas, no podrá beneficiarse de todas las funcionalidades que ésta ofrece.

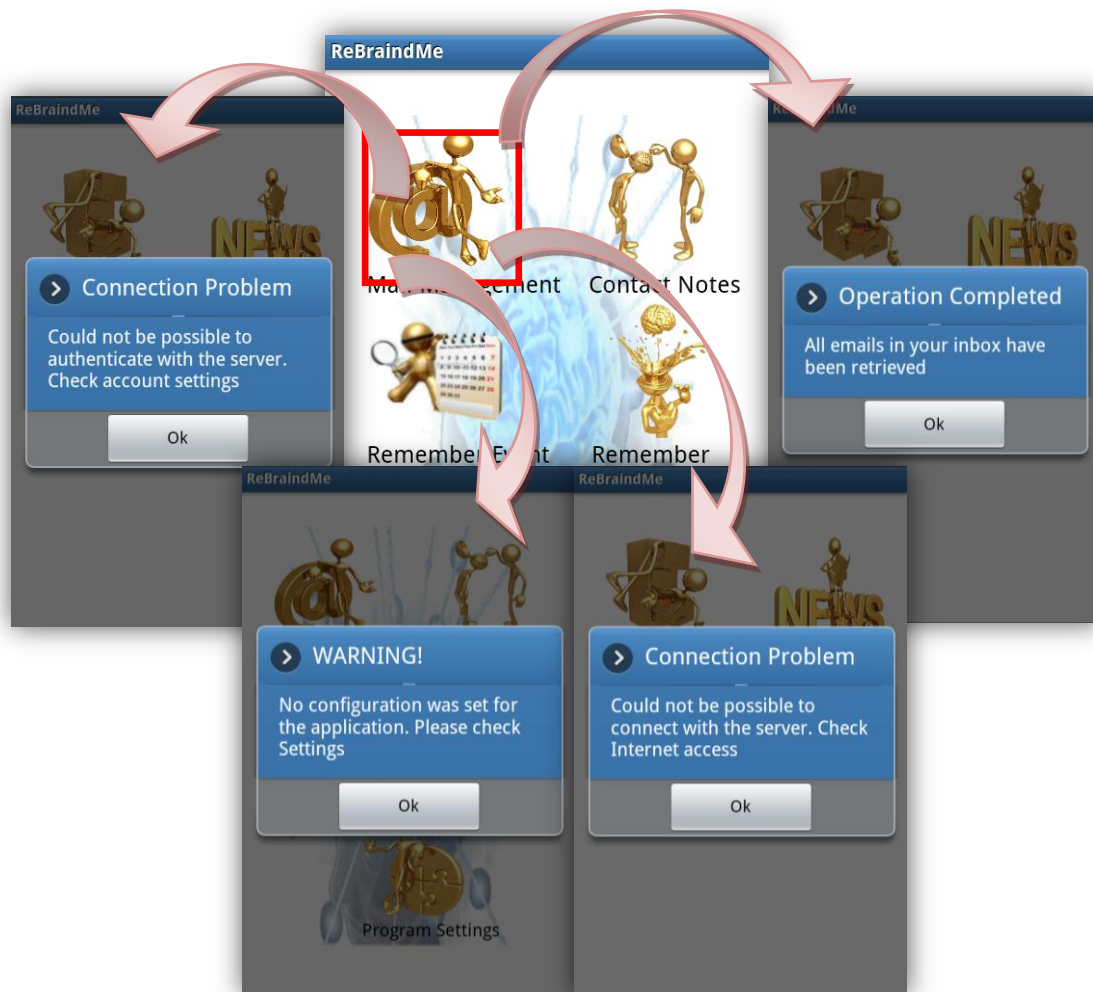


**Figura 35. Interfaz de configuración**

Esta interfaz se muestra simple con cuatro cuadros de texto en los cuales el usuario debe introducir las diferentes configuraciones de cuentas, permitiendo almacenarlas en la aplicación para que una vez guardadas, no haga falta volver a introducirlas cada vez que inicie la aplicación, a excepción de que el usuario desee cambiarlas.

Volviendo al menú principal, desde el mismo, el usuario puede navegar hacia la gestión de correo electrónico desde el botón de “Mail Management” el cual abrirá otro menú relacionado con el correo del usuario. De esta transición pueden ocurrir tres variaciones. La primera, en caso que no se haya configurado previamente la aplicación, ésta avisará al usuario mediante una ventana emergente comunicando que no existe una configuración inicial para la aplicación. Por otro lado, si durante la conexión y sincronización con el servidor ocurre un problema como un error de conexión o que los datos son incorrectos, una ventana emergente será desplegada para avisar al usuario. Por el contrario, si no ocurre ningún problema, aparecerá una ventana avisando de que se ha realizado la descarga de nuevos correos.





**Figura 36. Interfaces emergentes para el correo**

Estas ventanas permiten al usuario saber que está ocurriendo en la aplicación y cuál es el estado de ésta. Así, en caso de que ocurra algún error, el usuario tendrá información del mismo y en caso que pueda, tendrá datos suficientes para solucionarlo.

Tanto las ventanas de error, como de aviso y como de operación completada, permitirán seguir navegando al usuario al menú de correo electrónico. De esta manera el usuario puede seguir trabajando ya que por ejemplo, si desea utilizar la aplicación sin conexión a internet, solamente para ver sus correos descargados, podrá hacerlo sin ningún problema.

A pesar de ello, esto tendrá algunas restricciones, ya que el usuario no podrá enviar correos si no tiene conexión o si su configuración del servidor de correo no es correcta.

Para el caso que la aplicación haya terminado la descarga de manera satisfactoria, el usuario sabrá que puede acceder a su bandeja de entrada.

Una vez que el usuario accede al menú de gestión de correo electrónico se encontrará con cuatro botones bastante intuitivos. Los botones “Inbox” y “Unread Mail” permiten acceder tanto a los correos ya leídos como a los nuevos mostrando ambos la misma interfaz, como se puede apreciar en la siguiente figura. En esta interfaz se listan los correos del usuario, que si éste pulsa sobre cualquiera de ellos, se mostrará en una ventana el contenido de éste junto con cuatro botones que permiten responder al remitente del correo, responder a todos los participantes del correo, eliminar el correo, y por último enviar al calendario, en el cual el usuario rellenará un formulario con los datos necesarios para crear un evento en el calendario de Google. Hay que indicar que en caso de que se esté visualizando un correo que haya sido enviado al calendario, este botón cambiará de nombre a añadir otra vez al calendario, para que el usuario sea consciente de que el evento ya se envió.

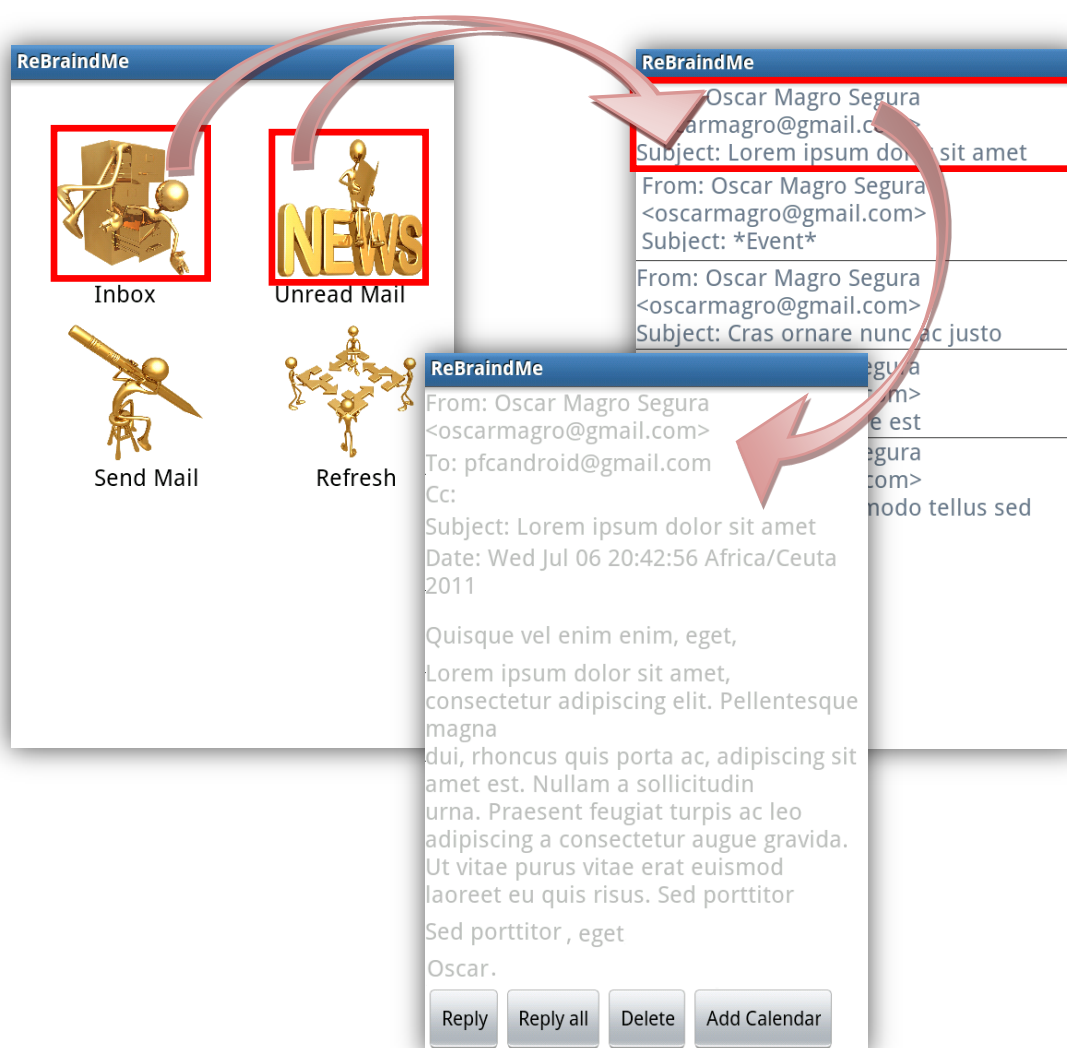


Figura 37. Interfaces Bandeja - Correo

**ReBraindMe**

From: Oscar Magro Segura  
<oscarmagro@gmail.com>  
To: pfcandroid@gmail.com  
Cc:  
Subject: Lorem ipsum dolor sit amet  
Date: Wed Jul 06 20:42:56 Africa/Ceuta 2011

Quisque vel enim enim, eget,  
Lorem ipsum dolor sit amet,  
consectetur adipiscing elit. Pellentesque magna  
dui, rhoncus quis porta ac, adipiscing sit  
amet est. Nullam a sollicitudin  
urna. Praesent feugiat turpis ac leo  
adipiscing a consectetur augue gravida.  
Ut vitae purus vitae erat euismod  
laoreet eu quis risus. Sed porttitor  
Sed porttitor , eget  
Oscar.

Reply Reply all Delete Add Calendar

**ReBraindMe**

From: Oscar Magro Segura  
<oscarmagro@gmail.com>  
To: pfcandroid@gmail.com  
Cc:  
Subject: \*Event\*  
Date: Wed Jul 06 22:17:25 Africa/Ceuta 2011

<Title>Cras scelerisque pharetra velit</Title>  
<Description>Phasellus quis quam  
congue mauris cursus consectetur.  
Phasellus  
id ligula id magna semper iaculis.  
Aenean fringilla nunc et nisl varius  
venenatis.</Description>  
<Location>Nullam sodales augue </Location>  
<Start>2011-07-11T12:00:00.000</Start>  
<End>2011-07-11T13:30:00.000</End>

Reply Reply all Delete Add Again to Calendar

**ReBraindMe**

Title

Initial Date DD / MM / YYYY

Initial Hour HH : MM : SS

Final Date DD / MM / YYYY

Final Hour HH : MM : SS

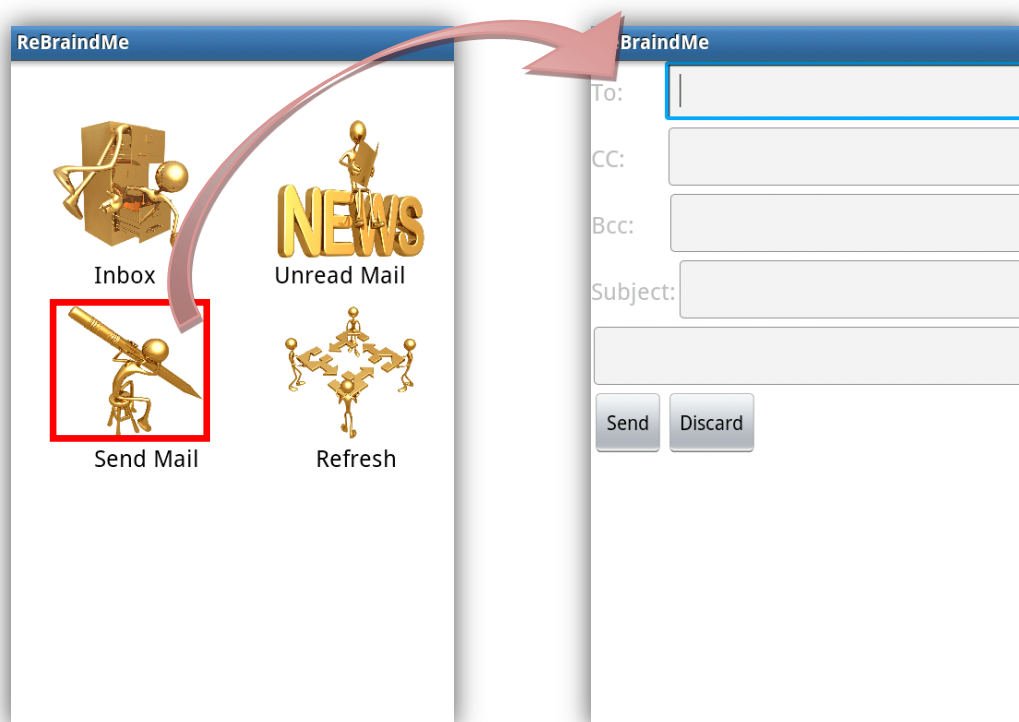
Location

Send Event Discard

Figura 38. Añadir Correo al Calendario

El botón de enviar correo permite al usuario redactar un nuevo mail mediante un formulario en el cual se deberá rellenar obligatoriamente como mínimo, el destinatario, ya que en caso contrario el correo no se enviará. Una vez con los campos rellenos, el usuario podrá enviar el mail mediante el botón “Send”, o si bien desea descartar los cambios bastará con pulsar sobre “Discard” y volveremos a la ventana anterior.

Esta misma interfaz se mostrará cuando el usuario decida responder a un correo, que a diferencia de este caso, automáticamente se rellenará el destinatario.

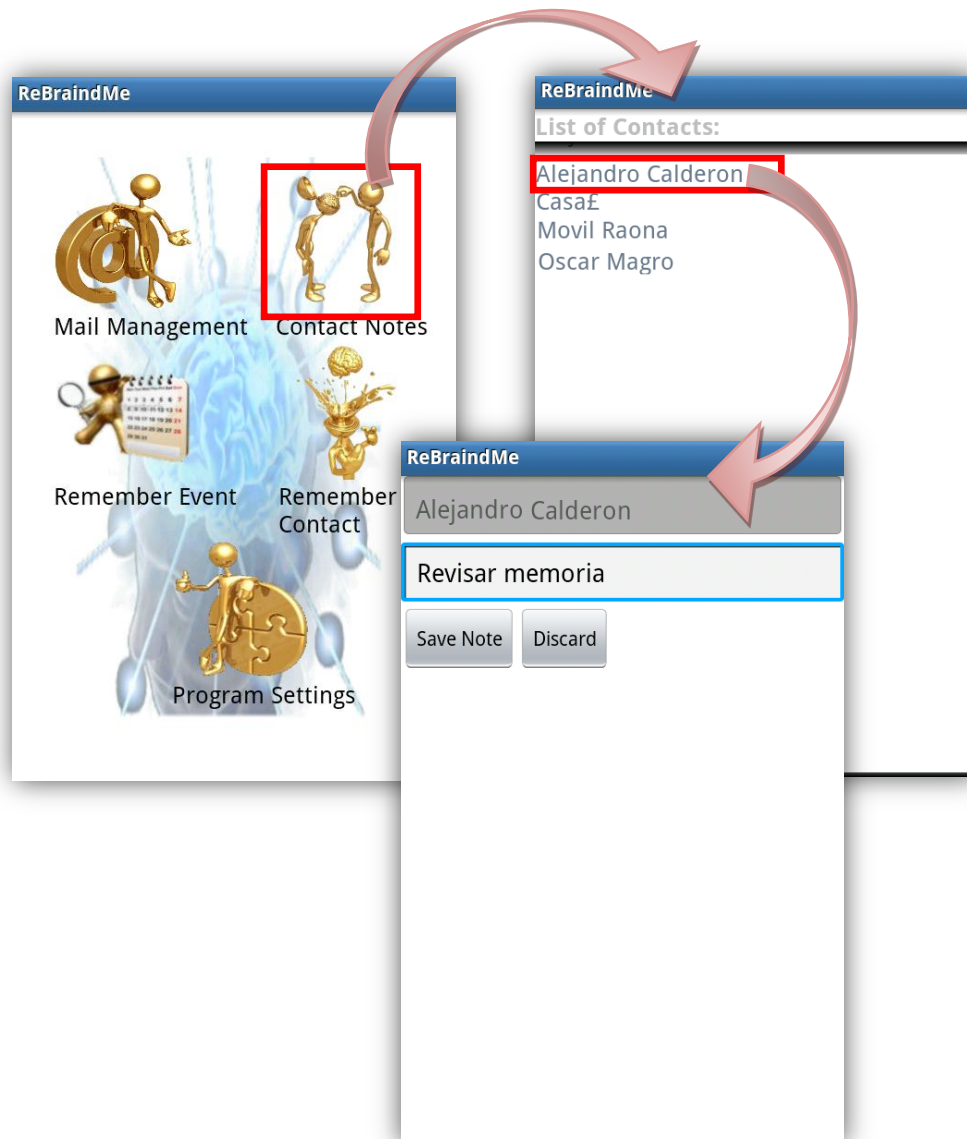


**Figura 39. Enviar correo electrónico**

Por último, dentro el menú de gestión de correo, el usuario puede encontrar un botón de refresco. Su intención es la de actualizar la bandeja de entrada de manera manual y de esta forma la aplicación vuelve a conectarse con el servidor, y en caso de que existan nuevos correos, éstos se descargarán.

Se podría haber implementado una actualización automática, pero como se pretende que el usuario también pueda utilizar la aplicación sin conexión a internet, de esta forma se evita tener procesos en memoria que estén ocasionando errores, aunque tratados, ya que al intentar conectarse con el servidor, esto generaría excepciones al no tener acceso a la red.

Otra funcionalidad a la que el usuario puede acceder desde el menú principal de la aplicación es la de añadir notas a los contactos. Al iniciar esta actividad se mostrará la lista de todos los contactos del usuario, que permitirá seleccionar al contacto al que se desea añadir la nota. Para ello, se abrirá un formulario en el cual se podrá introducir una descripción de la información que se desea asociar al contacto y guardarla en el sistema para más tarde ser consultada.



**Figura 40. Interfaces Añadir Nota Contacto**

Esta lista de contactos contendrá todos los que se encuentran en el dispositivo, por lo que aparecerán tanto los de la agenda como aquellos sincronizados desde otras aplicaciones como cuentas de correo, Facebook, y demás. Por lo tanto se presenta al usuario ordenada por nombre para facilitarle la tarea de búsqueda y selección.

Si por el contrario, en lugar de querer añadir una nota lo que el usuario desea es visualizar las notas que posee un contacto, éste podrá acceder a la lista de contactos desde el menú principal y ver las notas relacionadas con el contacto. Éstas se presentarán en forma de lista, con la descripción creada como texto. También se ofrece la posibilidad al usuario de eliminarlas, ya que es posible que una vez que haya realizado la tarea que tenía pendiente con ese contacto desee eliminar la nota del sistema, por lo que bastará con mantener pulsado sobre la nota durante unos segundos para que aparezca un menú contextual en el cual se dará la posibilidad de eliminar dicha nota.



Figura 41. Interfaces Visualizar Notas

Ya solo queda comentar la funcionalidad de visualizar los eventos. Para ello se accederá desde el menú principal, sobre el icono representado con un calendario. Esta acción mostrará en un listado todos los eventos almacenados que se han añadido al calendario en Google Calendar. Seleccionando un evento, el usuario podrá visualizar su contenido, en el que se indica el título, localización, descripción y las horas de inicio y final del evento.

A su vez, se ofrece la posibilidad de eliminar el evento mediante el botón de borrado. También se ha añadido otro botón mediante el cual el usuario puede enviar por correo electrónico este evento, de esta manera puede compartirlo con sus contactos, y si estos disponen de la aplicación, también se les añadirá al calendario.

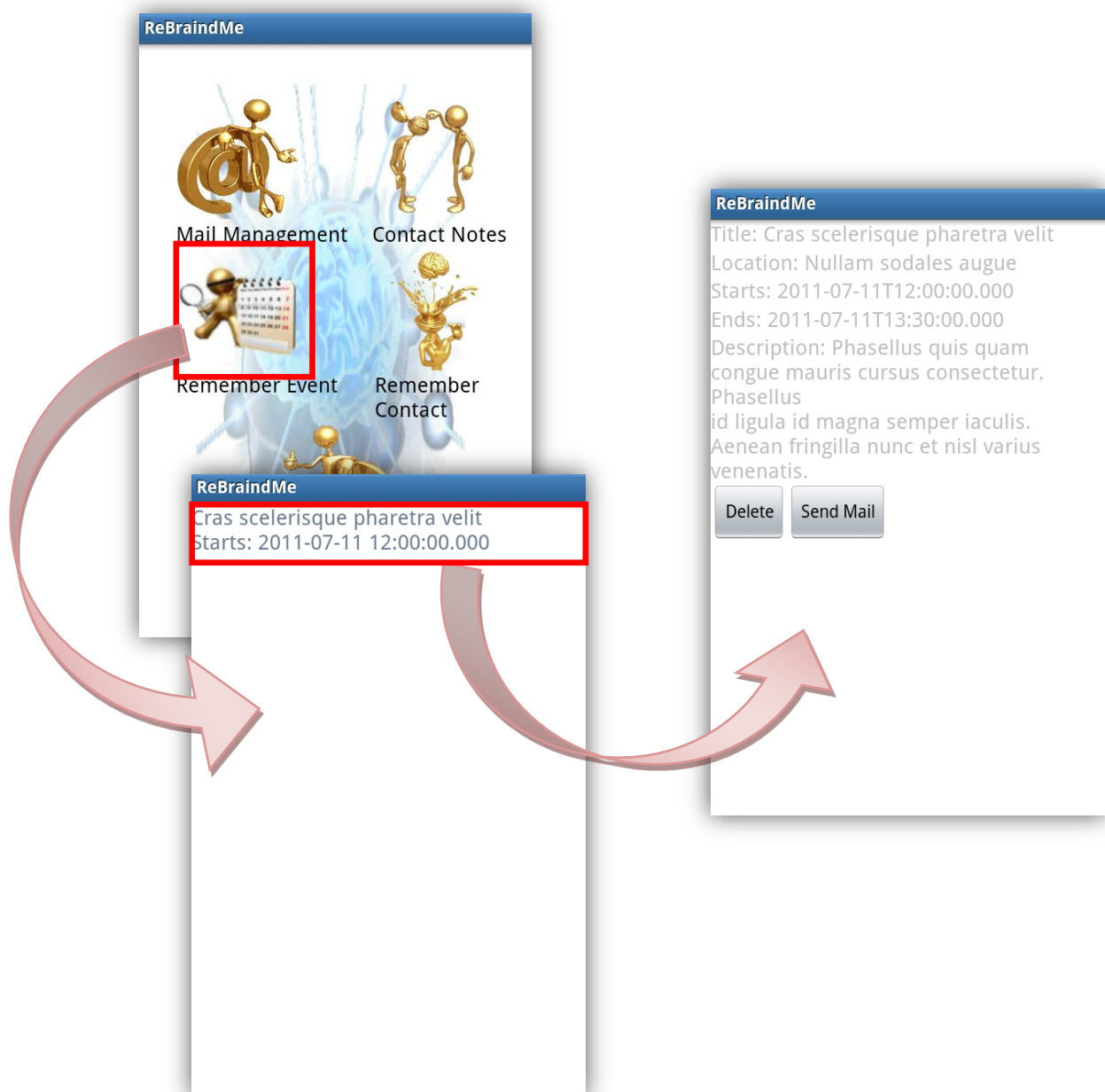
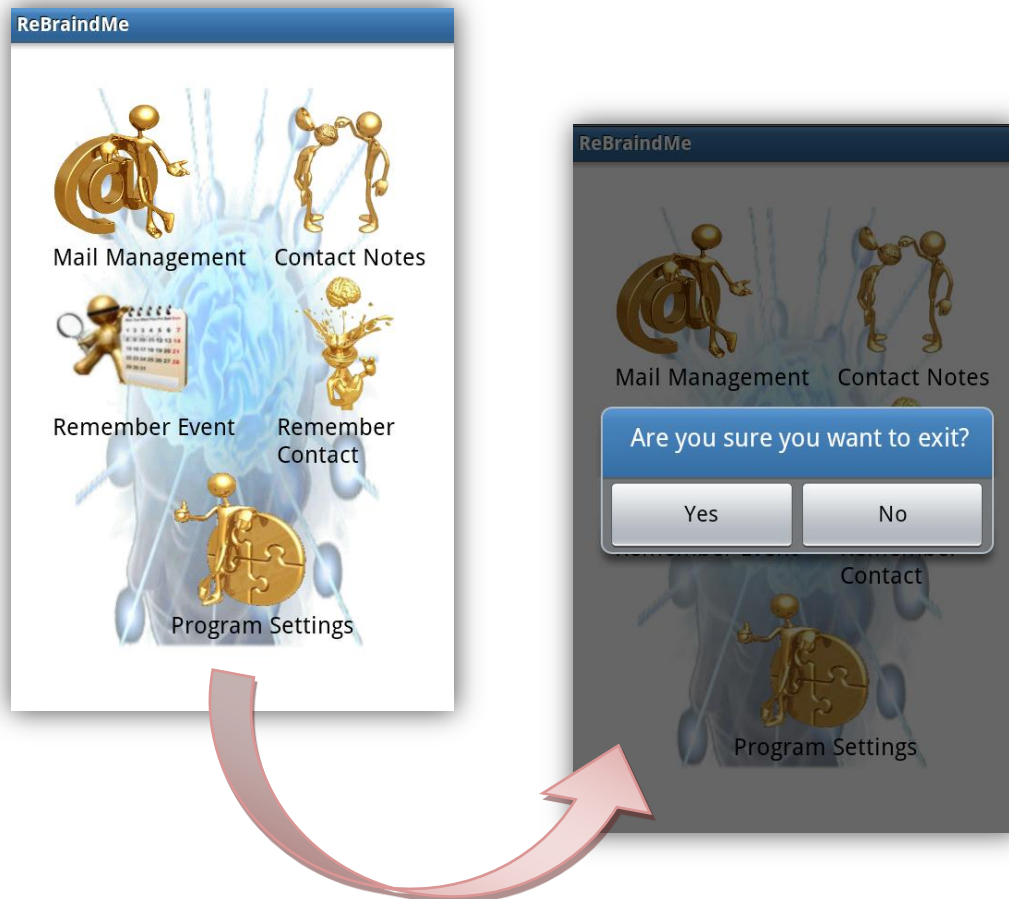


Figura 42. Interfaces Visualizar Eventos



Para terminar, si se desea finalizar la aplicación y que el usuario pueda salir de ella, bastará con usar el botón “back” del dispositivo. Esta acción lanzará una ventana emergente notificando al usuario que se dispone a salir de la aplicación, pudiendo aceptar o cancelar la acción.



**Figura 43. Interfaz Salida Aplicación**



## 4.4 Implementación

En esta sección se van a comentar los aspectos más relevantes respecto a la implementación de la aplicación, así como las consideraciones tomadas durante el desarrollo de la misma. No se pretende crear un manual de desarrollo extenso por lo que aspectos básicos de implementación relacionados con Android como puede ser crear actividades, pasar argumentos entre llamadas u otros temas triviales de esta plataforma pueden encontrarse en los tutoriales ofrecidos por la página de desarrollo de Android Android: .

Para exponer estos puntos se hará tomando como referencia el diagrama de componentes obtenido durante la fase de diseño, se dividirá la sección en seis partes.

### 4.4.1 Base de datos

Este componente es el encargado de gestionar el almacenamiento de los datos de la aplicación así como su uso. Para ello se ha utilizado el sistema gestor de base de datos relacional SQLite contenido en una biblioteca. Su funcionamiento es simple ya que no es un proceso independiente sino que se enlaza con el programa pasando a ser parte integral del mismo, de esta manera la aplicación utiliza las funcionalidades de SQL a través de llamadas simples reduciendo la latencia en el acceso a la base de datos. La manera en que se guarda el conjunto de la base de datos es como un único fichero estándar, el cual se bloquea al principio de cada transacción.

La forma de trabajar con los datos se realiza a través de un cursor que itera por las entradas de las diferentes tablas y con los identificadores de las entradas de las mismas.

Por lo tanto, lo primero que se debe hacer es definir la estructura de las tablas y las sentencias de creación de las mismas.

```
/**
 * Sentencias de creacion de tablas
 */
private static final String DATABASE_MAIL_CREATE = "create table mails (_id integer primary key autoincrement, "
    + "fromfield text not null, tofield text not null, cc text, "
    + "subject text, datefield text, body text, type text not null);";

private static final String DATABASE_EVENT_CREATE = "create table events (_id integer primary key autoincrement, "
    + "eventtitle text, eventlocation text, eventdescription text, "
    + "eventdayinitial text, eventdayfinal text);";

private static final String DATABASE_CONTACT_CREATE = "create table contacts (_id integer primary key autoincrement, "
    + "contactname text, contactnote text);";

private static final String DATABASE_USER_CREATE = "create table user (_id integer primary key autoincrement, "
    + "usermail text, passwordmail text, usercalendar text, passwordcalendar text);";
```

Figura 44. Sentencias Creación Tablas

Estas sentencias crearán las nuevas tablas en la base de datos. Para ello, se instancia a la base de datos y mediante comandos SQL se ejecutarán dichas sentencias. En la siguiente figura se muestra cómo realizar estos pasos.

```
private static final String DATABASE_NAME = "data";
private static final String DATABASE_TABLE = "mails";
private static final String DATABASE_EVENT_TABLE = "events";
private static final String DATABASE_CONTACT_TABLE = "contacts";
private static final String DATABASE_USER_TABLE = "user";

private static final int DATABASE_VERSION = 2;

private final Context mContext;

private static class DatabaseHelper extends SQLiteOpenHelper {

    DatabaseHelper(Context context) {
        super(context, DATABASE_NAME, null, DATABASE_VERSION);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(DATABASE_MAIL_CREATE);
        db.execSQL(DATABASE_EVENT_CREATE);
        db.execSQL(DATABASE_CONTACT_CREATE);
        db.execSQL(DATABASE_USER_CREATE);
    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) {
        Log.w(TAG, "Upgrading database from version " + oldVersion + " to "
            + newVersion + ", which will destroy all old data");
        db.execSQL("DROP TABLE IF EXISTS mails");
        onCreate(db);
    }
}
```

Figura 45. Ejecución Sentencias SQL

A la hora de crear una nueva entrada, se realizará mediante la sentencia de inserción de la base de datos, en el cual se le indica los valores que tomará el nuevo elemento y la tabla en la que se va a insertar la nueva entrada.

```

/**
 * Crea una nueva entrada en la tabla Mail a partir de un objeto Mail.
 * Si la operación es correcta devolverá el ID de la fila, en caso contrario
 * devolverá -1 indicando el fallo.
 *
 * @param mail
 *         mail que será creado
 * @return rowId or -1 si error
 */
public long createMail(Mail mail) {
    ContentValues initialValues = new ContentValues();

    initialValues.put(FROM, mail.getFrom());
    initialValues.put(TO, mail.getTo());
    initialValues.put(CC, mail.getCc());
    initialValues.put(SUBJECT, mail.getSubject());
    initialValues.put(SENT_DATE, mail.getDate());
    initialValues.put(BODY_MAIL, mail.getBody());
    initialValues.put(MAIL_TYPE, mail.getType());
    initialValues.put(MAIL_STRING, "From: ".concat(mail.getFrom()).concat("\n").concat("Subject: ").concat(mail.getSubject()));
    return mDb.insert(DATABASE_TABLE, null, initialValues);
}

```

Figura 46. Insertar Datos Tabla

A su vez, SQLite nos permite eliminar elementos de una tabla mediante su identificador.

```

/**
 * Elimina un registro correo a partir de su rowId
 *
 * @param rowId
 *         id del correo a borrar
 * @return true si borrado, false en caso contrario
 */
public boolean deleteMail(long table_id) {

    return mDb.delete(DATABASE_TABLE, TABLE_ID + "=" + table_id, null) > 0;
}

```

Figura 47. Eliminar entradas

Otra funcionalidad muy necesaria es la de poder consultar los datos y obtenerlos mediante consultas. Éstas permiten añadir condiciones como por ejemplo si se desean obtener los correos que ya han sido leídos.

```

/**
 * Devuelve un cursor sobre la lista de todos los correos no leídos de la tabla
 *
 * @return Cursor
 */
public Cursor fetchAllUnreadMails() {

    return mDb.query(DATABASE_TABLE, new String[] { TABLE_ID, FROM, TO, CC,
        SUBJECT, DATE_SENT, BODY_MAIL, MAIL_TYPE },
        MAIL_TYPE + "=0", null, null, null, null);
}

```

Figura 48. Consulta Tabla

Por último, otra característica que se ha utilizado es la de modificar una entrada dentro de la base de datos. Esto puede ocurrir por ejemplo para el caso de los correos si el usuario ha leído ese correo, se modifica su característica para así mostrarlo después en la bandeja correspondiente.

```
/**
 * Modifica el correo usando los datos indicados.
 *
 * @param table_id
 *         id del correo a modificar
 * @param mail
 *         valores del correo que se van a modificar
 *
 * @return true si se ha modificado correctamente, false en caso contrario
 */
public boolean updateMail(long table_id, Mail mail) {
    ContentValues args = new ContentValues();

    args.put(FROM, mail.getFrom());
    args.put(TO, mail.getTo());
    args.put(CC, mail.getCc());
    args.put(SUBJECT, mail.getSubject());
    args.put(SENT_DATE, mail.getDate());
    args.put(BODY_MAIL, mail.getBody());
    args.put(MAIL_TYPE, mail.getType());

    return mDb
        .update(DATABASE_TABLE, args, TABLE_ID + "=" + table_id, null) > 0;
}
```

Figura 49. Modificar entrada

### 4.4.2 Gestor de correo

Este elemento puede ser el más importante de la aplicación ya que la mayor parte de ésta hace uso de este componente. Para desarrollarlo se ha tenido que enfrentarse a varios problemas como se ha comentado a lo largo del documento.

En un principio se intentó utilizar el protocolo POP3 para conectarse al servidor y obtener los mensajes de la bandeja de entrada, pero esto ocasionaba errores en la lectura del contenido del mensaje para cuando se daba el caso que éste contenía un archivo adjunto. Esto se producía debido a que la librería utilizada de JavaMail adaptada para Android no soportaba este caso por lo que al final se decidió utilizar IMAP4.

Otro problema que se ha mencionado ha sido a la hora de autenticarse con el servidor. Ya se han comentado en puntos anteriores los diferentes métodos de autenticación que Android ofrece, pero debido a que a la hora de desarrollar la aplicación se trabajó con el emulador que proporcionaba para Eclipse, el método de autenticación que podría haber sido más correcto, el que utiliza el Account Manager de Android, no se ha podido utilizar al no estar habilitada esta opción dentro del emulador, por lo que al final se terminó utilizando el método basado en ClientLogin. El problema que presenta este método es que el usuario debe proporcionar las claves a la aplicación por lo que debe confiar en ella.

Para crear una conexión a través de IMAP, se debe instanciar una sesión en la cual se deben indicar sus propiedades. Para ello se utiliza la librería de Java.util.Properties, que permite crear propiedades y almacenar los valores. Se necesitará indicar el protocolo y a su vez el puerto por el que se va a realizar la conexión. Para el caso de Google y su servicio de Gmail, este puerto será el 993 para conexiones IMAP. A su vez hay que indicar que la conexión será segura mediante el uso de SSL y el uso de SocketFactory. En la siguiente imagen se muestra cómo deben rellenarse estas propiedades y así poder crear la sesión.

```
Properties props = System.getProperties();  
// Proveedor IMAP  
props.setProperty( "mail.imap.socketFactory.class",  
    "javax.net.ssl.SSLSocketFactory");  
props.setProperty( "mail.imap.socketFactory.fallback", "false");  
props.setProperty( "mail.imap.port", "993");  
props.setProperty( "mail.imap.socketFactory.port", "993");  
  
Session session = Session.getInstance(props);
```

Figura 50. Configurar Conexión IMAP

Una vez creada la sesión, se debe establecer la conexión. Utilizar ClientLogin es algo muy sencillo ya que basta con indicarle las credenciales y la dirección a la que se va a conectar con la sesión y el protocolo.

```
try{

    //Se crea el almacen y se realiza la conexión a través de éste
    Store store = session.getStore("imaps");
    store.connect("imap.gmail.com", Constants.USERNAME_MAIL, Constants.PASSWORD_MAIL);
    Folder folder = store.getFolder("INBOX");
    folder.open(Folder.READ_WRITE);

    // Se obtienen los mensajes.
    FlagTerm ft = new FlagTerm(new Flags(Flags.Flag.SEEN), false);
    Message[] mensajes = folder.search(ft);
```

Figura 51. Conexión y obtención de mensajes

Una vez abierta la conexión, IMAP permite acceder a cualquier carpeta dentro del servidor de correo del usuario. Como de momento sólo interesa la bandeja de entrada, se obtendrá la carpeta “INBOX” con modo lectura escritura. A su vez, sólo tiene sentido obtener los mensajes nuevos que no han sido leídos, por lo que el flag “SEEN” se pondrá a false. Obtenida la carpeta se obtendrán los mensajes almacenándolos en una matriz para trabajar con ellos. Esto permitirá acceder a los elementos del mensaje y almacenarlos en la base de datos de la aplicación.

Otra operación que realiza este gestor es la de enviar correos electrónicos. Para ello en lugar de utilizar una conexión IMAP se utiliza SMTP. A la hora de establecer la conexión, no es muy diferente a lo realizado para conectarse y descargarse correos. Se debe establecer una sesión de conexión para después crear un mensaje de tipo MimeMessage, el cual se deberá rellenar indicando destinatarios, tema, el cuerpo y demás elementos.

```
// Propiedades de la conexión
Properties props = new Properties();
props.setProperty("mail.smtp.host", "smtp.gmail.com");
props.setProperty("mail.smtp.starttls.enable", "true");
props.setProperty("mail.smtp.port", "587");
props.setProperty("mail.smtp.user", Constants.USERNAME_MAIL);
props.setProperty("mail.smtp.auth", "true");

// Preparamos la sesion
Session session = Session.getDefaultInstance(props);

// Construimos el mensaje
MimeMessage message = new MimeMessage(session);
```

Figura 52. Conexión SMTP

Una vez formado el mensaje se debe enviar creando un paquete de transporte. Éste se conectará con el servidor con unas credenciales y se enviará a los destinatarios añadidos al mismo.

```
// Lo enviamos.  
Transport t = session.getTransport("smtp");  
t.connect(Constants.USERNAME_MAIL, Constants.PASSWORD_MAIL);  
t.sendMessage(message, message.getAllRecipients());  
// Cierre.  
t.close();
```

Figura 53. Enviar Correo

En el caso de que la dirección esté mal formada se obtendrá una excepción avisando del error, por lo que es importante revisar que cumple la estructura de una cuenta de correo, es decir ejemplo@ejemplo.com.

### 4.4.3 Gestor de Eventos

En esta sección se describirá como se debe interactuar con el servicio de Google Calendar a la hora de conectarse con el calendario del usuario y manejar eventos. El primer paso a realizar es conectarse al servicio y obtener el calendario del usuario. Para ello hay que abrir la conexión con el servidor utilizando las credenciales del usuario, obtener un token, que se utilizará más tarde para las peticiones al servidor, y construir la dirección para trabajar con el calendario.

```
public int authenticate() {

    int code = 0;
    HttpTransport transport = GoogleTransport.create();
    GoogleHeaders headers = (GoogleHeaders) transport.defaultHeaders;
    headers.setApplicationName("Google-RebraindMe/1.0");
    headers.gdataVersion = "2";
    AtomParser parser = new AtomParser();
    parser.namespaceDictionary = Namespace.DICTIONARY;
    transport.addParser(parser);

    ClientLogin authenticator = new ClientLogin();
    authenticator.authTokenType = TOKEN_TYPE;
    authenticator.username = Constants.USERNAME_CALENDAR;
    authenticator.password = Constants.PASSWORD_CALENDAR;
    authenticator.accountType = ACCOUNT_TYPE;

    try {

        authenticator.authenticate().setAuthorizationHeader(transport);
        CalendarUrl url = CalendarUrl
            .forDefaultPrivateFullEventFeed(Constants.USERNAME_CALENDAR);
```

Figura 54. Autenticación Google Calendar

Esta conexión necesitará de una capa http de transporte que en lugar de tener que construirla desde cero, se utilizará aquella proporcionada por Google dentro de sus librerías para el cliente. Se deberá configurar sus cabeceras y como se está trabajando con Atom XML, se definirá un parser de tipo Atom para los mensajes intercambiados entre el cliente y el servidor.

En cuanto a la autenticación, se vuelve a utilizar ClientLogin, por lo que se necesitarán las credenciales facilitadas por el usuario. A su vez se deberá indicar al servidor que el tipo de token será “cl” de ClientLogin y que la cuenta será de tipo “Hosted\_or\_Google” lo que significa que primero se intentará obtener las credenciales para una cuenta almacenada por otro servidor y en caso de que falle se obtendrán para la cuenta Google. Es recomendable utilizar esta opción si no se está seguro del tipo de cuenta de la que se quiere obtener la autenticación. Una vez todo configurado, al realizar la autenticación con el servidor Google, el token se asociará al transporte.



Para construir la url del calendario se debe decidir con cual se va a trabajar, por lo que se ha considerado que se utilizará el calendario principal del usuario, sabiendo que siempre existirá y su dirección tendrá la siguiente estructura.

`https://www.google.com/calendar/feeds/<username>/private/full`

```
public static CalendarUrl forDefaultPrivateFullEventFeed(String user) {
    return forEventFeed(user, "private", "full");
}

public static CalendarUrl forEventFeed(
    String userId, String visibility, String projection) {
    CalendarUrl result = forRoot();
    result.pathParts.add(userId);
    result.pathParts.add(visibility);
    result.pathParts.add(projection);
    return result;
}

private static CalendarUrl forRoot() {
    return new CalendarUrl(ROOT_URL);
}

public static final String ROOT_URL =
    "https://www.google.com/calendar/feeds";
```

Figura 55. Creación URL calendario del usuario

Con esta dirección y el token de autenticación, la aplicación ya es capaz de conectarse al calendario del usuario y empezar a trabajar con evento.

Si se recuerda lo explicado en el apartado 3.5, los eventos tienen un formato xml como el siguiente,

```
<entry xmlns='http://www.w3.org/2005/Atom'
  xmlns:gd='http://schemas.google.com/g/2005'>
  <category scheme='http://schemas.google.com/g/2005#kind'
    term='http://schemas.google.com/g/2005#event'></category>
  <title type='text'>Tennis with Beth</title>
  <content type='text'>Meet for a quick lesson.</content>
  <gd:transparency
    value='http://schemas.google.com/g/2005#event.opaque'>
  </gd:transparency>
  <gd:eventStatus
    value='http://schemas.google.com/g/2005#event.confirmed'>
  </gd:eventStatus>
  <gd:where valueString='Rolling Lawn Courts'></gd:where>
  <gd:when startTime='2006-04-17T15:00:00.000Z'
    endTime='2006-04-17T17:00:00.000Z'></gd:when>
</entry>
```

Figura 56. Ejemplo Evento Calendario Google

Ahora bien, para implementación en Java, Google propone unas funciones en sus librerías clientes que permiten crear eventos y añadirlos al calendario de manera rápida únicamente instanciando el evento y rellenando sus atributos tal y como muestra la siguiente captura.

```
URL postUrl =
    new URL("https://www.google.com/calendar/feeds/jo@gmail.com/private/full");
CalendarEventEntry myEntry = new CalendarEventEntry();

myEntry.setTitle(new PlainTextConstruct("Tennis with Beth"));
myEntry.setContent(new PlainTextConstruct("Meet for a quick lesson.));

DateTime startTime = DateTime.parseDateTime("2006-04-17T15:00:00-08:00");
DateTime endTime = DateTime.parseDateTime("2006-04-17T17:00:00-08:00");
When eventTimes = new When();
eventTimes.setStartTime(startTime);
eventTimes.setEndTime(endTime);
myEntry.addTime(eventTimes);

// Send the request and receive the response:
CalendarEventEntry insertedEntry = myService.insert(postUrl, myEntry);
```

Figura 57. Creación de un Evento en Java

En el caso del desarrollo de la aplicación realizado para Android, las librerías de Google no ofrecen estas funcionalidades por lo que es necesario crearse primero la estructura del evento, que tendrá un título, una descripción y dos objetos que representarán el lugar y la fecha del evento.

```
public int createEvent(String title, String content, String startTime, String endTime,
    String location){
    int code = 0;
    try{
        WhenCalendar eventTimes = new WhenCalendar();
        WhereCalendar where = new WhereCalendar();
        DateTime start = DateTime.parseRfc3339(startTime);
        DateTime end = DateTime.parseRfc3339(endTime);
        eventTimes.setStartTime(start);
        eventTimes.setEndTime(end);

        where.setValueString(location);

        event.setTitle(title);
        event.setContent(content);
        event.setWhen(eventTimes);
        event.setWhere(where);
        return code;
    }catch(Exception e){
        code = -1;
        return code;
    }
}

public class WhenCalendar{
    @Key("@startTime")
    private DateTime startTime;

    @Key("@endTime")
    private DateTime endTime;
}

public class CalendarEvent {
    @Key
    private String title;

    @Key
    private String content;

    @Key("gd:when")
    private WhenCalendar when;

    @Key("gd:where")
    private WhereCalendar where;
}

public class WhereCalendar {
    @Key("@valueString")
    private String valueString;
}
```

Figura 58. Creación Evento Android

Es importante fijarse en las propiedades “Key” y sus valores para los atributos del evento, ya que si éstas no se indican, el evento no se creará correctamente ya que al transformar el objeto a xml, el servicio de Google no será capaz de establecer cual serán los campos del nuevo evento.

Creada la conexión con el servidor y el evento, obtenido el token de autenticación, y la dirección del calendario, ya sólo queda ver cómo se debe insertar el nuevo evento en el calendario del usuario.

Se construirá un mensaje de petición en el cual se creará un contenido XML para agregar el evento y se indicará la dirección del calendario. Una vez preparado el mensaje se enviará la petición y se obtendrá la respuesta del servidor.

```
public CalendarEvent executeInsert(HttpTransport transport, CalendarUrl url)
    throws IOException {
    HttpRequest request = transport.buildPostRequest();
    request.url = url;
    AtomContent content = new AtomContent();
    content.namespaceDictionary = Namespace.DICTIONARY;
    content.entry = this;
    request.content = content;
    return RedirectHandler.execute(request).parseAs(getClass());
}
```

**Figura 59. Enviar un Evento a GoogleCalendar**

Esta respuesta variará dependiendo del resultado de la ejecución de la petición. Si todo ha ido bien el evento se habrá añadido al calendario del usuario pero pueden ocurrir diferentes excepciones durante la ejecución, ya sea porque está mal construida la dirección y no se pueda resolver, o bien por falta de conexión a internet o problemas con la autenticación. Todos estos errores serán identificados por un código que se obtendrá con la respuesta del servidor.

De entre todos estos códigos, existe uno en particular que debe tratarse pues puede ocurrir de manera regular. Se trata del código 302, el cual indica que el servidor se ha movido o que la petición debe redirigirse a otra dirección url, por lo que debe capturarse esta excepción, y utilizando la misma sesión con las mismas credenciales se debe volver a realizar la petición a la nueva dirección indicada en el mensaje de respuesta del servidor.

```

static HttpResponse execute(HttpRequest request) throws IOException {
    try {
        return request.execute();
    } catch (HttpResponseException e) {
        if (e.response.statusCode == 302) {
            GoogleUrl url = new GoogleUrl(e.response.headers.location);
            request.url = url;
            new SessionInterceptor(request.transport, url);
            e.response.ignore(); // force the connection to close
            return request.execute();
        } else {
            throw e;
        }
    }
}

```

**Figura 60.** Captura excepción de redirección url

Con estas pautas indicadas, es fácil interactuar con el servicio de Google para manejar eventos, y a pesar de que no se disponen de las mismas funcionalidades para un desarrollo Java y uno Android, siguiendo los pasos explicados, se pueden suplir estas diferencias. Aun así, Google no tardará en ceder librerías exclusivas para Android que faciliten los desarrollos, ya que cada vez aparecen más desarrollos para esta plataforma que interactúa con sus servicios.

### 4.4.4 Gestor de Notas

Este componente puede ser, de entre todos, el más sencillo de desarrollar ya que tiene un funcionamiento más simple y únicamente utiliza las librerías de Android e interactúa con el propio dispositivo.

El funcionamiento de esta parte de la aplicación se basa en obtener los contactos del usuario almacenados en el dispositivo y mostrarlos en una lista para poder ser seleccionados y crear notas sobre ellos.

Comprendida la idea básica, el primer paso es recuperar los contactos del dispositivo. Para ello obtenemos un “*ContentResolver*”, que no es más que un contenedor de información sobre el cual se puede almacenar y acceder a la información que posee desde la aplicación. Se accede a la lista de los contactos almacenándolos en el contenedor mediante una consulta en la que se le indica la dirección dónde se almacenan los contactos y así obtener un cursor para moverse por los datos.

```
private void fillData(){

    ContentResolver cr = getContentResolver();
    Cursor cur = cr.query(ContactsContract.Contacts.CONTENT_URI, null,
        null, null, ContactsContract.Contacts.DISPLAY_NAME );
    int i = 0;
    if (cur.getCount() > 0) {
        String[] names = new String[cur.getCount()];
        while (cur.moveToNext()) {
            String id = cur.getString(cur
                .getColumnIndex(ContactsContract.Contacts._ID));
            String name = cur
                .getString(cur
                    .getColumnIndex(ContactsContract.Contacts.DISPLAY_NAME));
            names[i] = name;
            i++;
        }

        setListAdapter(new ArrayAdapter<String>(this,
            R.layout.contacts_row, names));
    }
}
```

Figura 61. Listar contactos usuario

Teniendo el cursor, basta con recorrer el contenedor e ir obteniendo el identificador y el nombre del contacto para mostrarlos en la interfaz.

Para crear una nota y almacenarla en el sistema se debe interactuar con la base de datos que se ha creado. El primer paso es instanciarla y abrirla, ya que si no está abierta se producirá una excepción. Después se obtendrán los datos de la nota que se desea crear y se llamará al método implementado para almacenar la nueva nota, que se compondrá de una consulta “Insert” con los parámetros que recibe. Por último, se debe cerrar la base de datos ya que si no se queda el recurso abierto y al ser una aplicación con diversas actividades que utilizan la misma base de datos, se pueden producir excepciones al tener el recurso ocupado o incluso perder la integridad de los datos.

```
private void saveNote() {
    mDbHelper = new RebraindmeDataBase(this);
    mDbHelper.open();

    EditText body_field = (EditText) findViewById(R.id.note_body);
    String body = body_field.getText().toString();

    mDbHelper.createContact(contact_name, body);
    mDbHelper.close();
}
```

Figura 62. Insertar nueva nota en el sistema

Si por el contrario en lugar de querer ver las notas almacenadas para un contacto se desea mostrar las que éste tiene almacenadas, se deberá abrir la base de datos al igual que en el caso anterior, y realizar una consulta que nos devolverá un cursor apuntando a los elementos devueltos por ésta, que usaremos para obtener los datos necesarios para rellenar la nota que se le mostrará al usuario mediante la interfaz.

```
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.contact_notes);
    mDbHelper = new RebraindmeDataBase(this);
    mDbHelper.open();
    Bundle extras = getIntent().getExtras();
    contact_name = extras.getString(RebraindmeDataBase.CONTACT_NAME);
    ((TextView) findViewById(R.id.contact_name_note)).setText
        (R.string.contact_note_intro+" "+contact_name);

    fillData();
    registerForContextMenu(getListView());
    mDbHelper.close();
}

private void fillData() {
    cursor = mDbHelper.fetchAllContact(contact_name);
    startManagingCursor(cursor);
    String[] from = new String[]{RebraindmeDataBase.CONTACT_NOTE};
    int[] to = new int[]{R.id.note_row1};
    SimpleCursorAdapter notes = new SimpleCursorAdapter
        (this, R.layout.note_row, cursor, from, to);
    setListAdapter(notes);
}
```

Figura 63. Mostrar una nota

Por último, queda ver cómo se puede eliminar una nota del sistema para cuando ya no sea necesario el recordatorio. Para ello se ha creado un menú contextual que aparecerá cuando se mantenga pulsada la nota. Lo que hará será capturar el identificador y abrir una conexión con la base de datos y lanzar la consulta de borrado sobre ese elemento para luego cerrar la conexión y el menú.

```
@Override
public void onCreateContextMenu(ContextMenu menu, View v,
    ContextMenuInfo menuInfo) {
    super.onCreateContextMenu(menu, v, menuInfo);
    menu.add(0, DELETE_ID, 0, R.string.delete_contact_note);
}

@Override
public boolean onContextItemSelected(MenuItem item) {
    switch(item.getItemId()) {
        case DELETE_ID:
            AdapterContextMenuInfo info = (AdapterContextMenuInfo) item.getMenuInfo();
            mDbHelper.open();
            mDbHelper.deleteContact(info.id);
            fillData();
            mDbHelper.close();
            return true;
    }
    return super.onContextItemSelected(item);
}
```

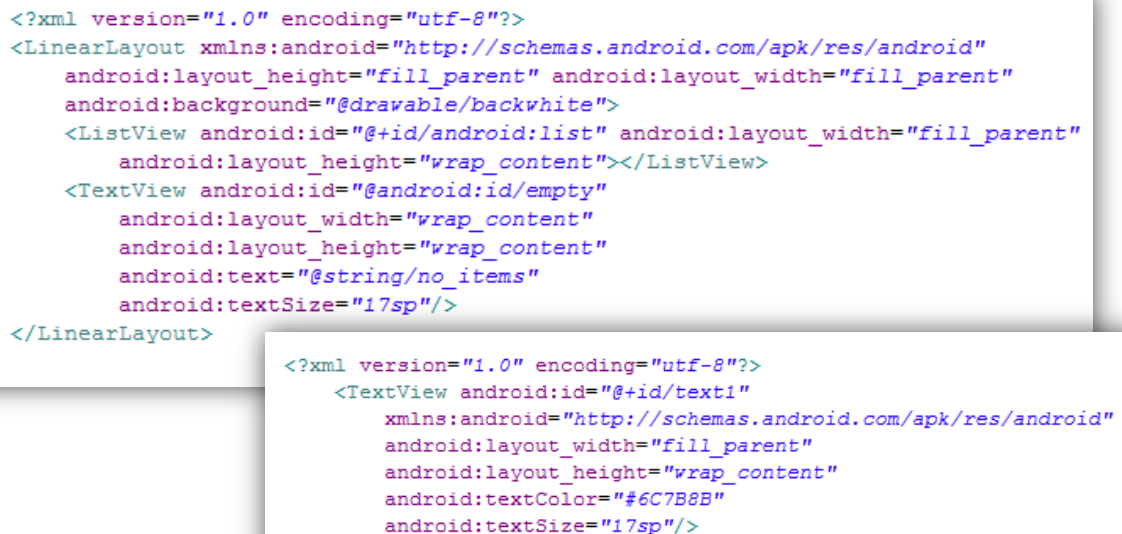
Figura 64. Eliminar nota del sistema

Debe quedar claro que estos procedimientos de ataque a la base de datos son idénticos para almacenar, consultar y borrar cualquier otro elemento de nuestra aplicación, a excepción de la llamada a la consulta que deberá cambiarse por la que proceda.

### 4.4.5 Interfaz de Usuario

La idea de este punto es comentar como realizar algunas interfaces y como capturar alguno de los eventos de las mismas. Como se ha explicado a lo largo del documento, las interfaces en Android se realizan mediante ficheros con etiquetas xml que definen elementos a los cuales se darán funcionalidades mediante el uso de código. No se van a explicar todas las interfaces realizadas ya que muchas de ellas comparten mismos elementos y lo que interesa de esta sección es conocer los se pueden realizar las acciones menos triviales para un desarrollo Android.

A continuación se comenta como se debe crear una lista de elementos. Esto se ha usado por ejemplo para listar los correos o los contactos. Para ello se necesitarán dos elementos xml, uno de ellos que se corresponderá al elemento que se va a mostrar y otro al contenedor de estos elementos. El fichero que representará al elemento contendrá un View de tipo texto para almacenar el valor. Por otro lado se tendrá un fichero representando a la lista donde se tendrá un Layout que agrupará a un View de tipo lista que contendrá a los elementos descritos anteriormente y otro View de tipo texto que se mostrará para el caso que no existan elementos mostrando un mensaje.



```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_height="fill_parent" android:layout_width="fill_parent"
    android:background="@drawable/backwhite">
    <ListView android:id="@+id/android:list" android:layout_width="fill_parent"
        android:layout_height="wrap_content"></ListView>
    <TextView android:id="@android:id/empty"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/no_items"
        android:textSize="17sp"/>
</LinearLayout>
```

```
<?xml version="1.0" encoding="utf-8"?>
<TextView android:id="@+id/text1"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:textColor="#6C7B8B"
    android:textSize="17sp"/>
```

Figura 65. XML Lista de elementos



En el siguiente ejemplo se muestra cómo se rellena la lista de correos no leídos. Para ello se obtienen los correos mediante una consulta y se crean dos matrices. La primera indicará los campos que se quieren mostrar en la lista y la segunda los campos en los que se desea mostrar esos elementos, en otras palabras el elemento creado en el fichero xml que corresponde al elemento de la lista. Con esto se crea un cursor de tipo “*SimpleCursorAdapter*” que se encargará de leer todos los elementos e irlos añadiendo a la lista.

```
private void fillData() {
    cursor = mDbHelper.fetchAllUnreadMails();
    startManagingCursor(cursor);

    String[] from = new String[]{RebraindmeDataBase.MAIL_STRING};

    int[] to = new int[]{R.id.text1};

    SimpleCursorAdapter mails = new SimpleCursorAdapter
        (this, R.layout.mail_row, cursor, from, to);
    setListAdapter(mails);
}
```

Figura 66. Rellenar una lista

Una interfaz que engloba tanto botones, como cuadros de texto y etiquetas puede ser la de crear un correo. La primera parte del fragmento de código se corresponde al campo del destinatario del correo. Se ha añadido un View de tipo texto para representar el literal y otro de tipo edición que corresponde a la entrada de texto en la que el usuario puede escribir. Para que estuvieran en el mismo plano horizontal se han rodeado por un Layout horizontal que a su vez lo rodea otro vertical, ya que si se observa la interfaz, ésta presenta los campos del mensaje uno debajo de otro. Se puede ver que existe una etiqueta ScrollView, esta se ha añadido debido a que el usuario es capaz de introducir varias líneas de texto en el cuerpo del correo por lo que de esta manera se permitirá al usuario desplazarse por la ventana mediante scroll.

En la segunda parte de código se ven como se han creado los botones tanto de envío como de descarte del correo. Al aparecer en horizontal uno con otro, se ha rodeado por otro Layout. Si se fija detenidamente en el código se puede ver que no está indicado el atributo de orientación horizontal. Esto es debido a que por defecto, un Layout muestra sus elementos de manera horizontal, por lo que si no se indica nada, el atributo orientación tomará este valor. En otras interfaces en lugar de usar un elemento de tipo “*Button*” se ha utilizado uno de “*ImageButton*” que es prácticamente igual a excepción de que este último contiene un atributo que permite agregar una imagen de fondo.

```

<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical" android:background="@drawable/backwhite"
    android:layout_width="fill_parent" android:layout_height="fill_parent"
    android:clickable="false" android:fitsSystemWindows="false"
    android:isScrollContainer="false">
    <ScrollView android:id="@+id/ScrollView01"
        android:layout_width="fill_parent" android:layout_height="fill_parent">
        <LinearLayout android:orientation="vertical"
            android:layout_height="wrap_content" android:layout_width="fill_parent">
            <LinearLayout android:orientation="horizontal"
                android:layout_height="wrap_content" android:layout_width="fill_parent">
                <TextView android:text="@string/to_mail" android:id="@+id/to_text"
                    android:layout_width="wrap_content" android:layout_height="wrap_content"
                    android:textSize="17sp"></TextView>
                <EditText android:text="" android:id="@+id/to_edit"
                    android:layout_width="fill_parent" android:layout_height="wrap_content"
                    android:layout_marginLeft="29sp"></EditText>
            </LinearLayout>
        </LinearLayout>
        <LinearLayout android:id="@+id/LinearLayout01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
            <Button android:id="@+id/Send" android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/send_button"></Button>

            <Button android:id="@+id/Discard"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/discard_button"></Button>
        </LinearLayout>
    </ScrollView>
</LinearLayout>

```

Figura 67. Código interfaz de creación de correo

Otro aspecto relevante puede ser el de crear diálogos. Esto se usa para avisar al usuario de que ha ocurrido algo en la aplicación, algunos ejemplos dentro de este desarrollo pueden ser cuando se ha producido un error al conectar con el servidor de correo o al salir de la aplicación.

En el siguiente ejemplo se muestra como se ha creado una ventana emergente en el caso de que los datos de un evento sean erróneos y se produzca un error al enviarse al calendario. Para crear esta ventana hay que instanciar un objeto de tipo “*AlertDialog*” y añadirle el título y la descripción. Dentro de la ventana también se pueden tantos botones como se desee, para este caso se crea uno, el cual cierra la ventana cuando es pulsado por el usuario.

```

new AlertDialog.Builder(this)
    .setTitle(R.string.error_title)
    .setMessage(R.string.error_calendar_description)
    .setPositiveButton("Ok", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface dialog, int which) {
            dialog.dismiss();
        }
    }).show();

```

Figura 68. Crear ventana emergente

Por último, otro dato interesante en la implementación de la interfaz ha sido la de capturar el evento que se genera cuando el usuario pulsa sobre la tecla de retorno en el dispositivo móvil. Si esta no es capturada la acción estándar es volver a la actividad anterior sin realizar ninguna otra acción, ya sea cerrar un recurso compartido o enviar un objeto o lanzar una ventana. Para el desarrollo de la aplicación se ha tenido que capturar en algunas ocasiones como puede ser la siguiente en la cual se genera cuando el usuario quiere salir de la aplicación, y según esta implementación, cuando el usuario va a salir se le informa de que está saliendo de la misma con una ventana emergente en el cual se le da la opción de confirmar la salida o cancelarla.

Para capturar estos eventos basta con sobrescribir el método “*onKeyDown*” tal y como muestra el ejemplo.

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    //Manejar el botón Back
    if(keyCode == KeyEvent.KEYCODE_BACK) {
        //Preguntar si desea salir con ventana emergente
        new AlertDialog.Builder(this)
            .setIcon(android.R.drawable.ic_dialog_alert)
            .setMessage("Are you sure you want to exit?")
            .setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                public void onClick(DialogInterface dialog, int which) {
                    //Terminar la actividad
                    MainMenu.this.finish();
                }
            })
            .setNegativeButton("No", null)
            .show();

        return true;
    }
    else {
        return super.onKeyDown(keyCode, event);
    }
}
}
```

Figura 69. Capturar tecla Back

# Capítulo 5

## Planificación y Presupuesto

### 5.1 Planificación

En esta sección dentro del capítulo se va a establecer la duración de las distintas tareas y actividades realizadas a lo largo del proyecto. Esta planificación se definirá en base a la herramienta del Diagrama de Gantt en la cual se representará gráficamente la duración de cada etapa o actividad a partir de una fecha comienzo y una de fin en base a un calendario. Este diagrama se representa en la imagen de la siguiente página.

En la siguiente tabla se muestran las actividades realizadas a lo largo del proyecto estableciendo un comienzo y un final que darán lugar a la entrada de datos del diagrama. Hay que tener en cuenta que el desarrollo del proyecto ha tenido que combinarse con otras tareas ajenas al proyecto por lo que la dedicación al mismo no ha sido total.

Tarea	Duración	Fecha de Inicio	Fecha de Fin
Estudio Viabilidad	10 días	01/06/10	13/06/10
Documentación Android	36 días	14/06/10	02/08/10
Fase de Análisis	45 días	03/08/10	04/10/10
Fase de Diseño	29 días	05/10/10	12/11/10
Fase de Implementación	150 días	15/11/10	10/06/11
Pruebas de la Aplicación	25 días	23/05/11	24/06/11
Documentación del Desarrollo	130 días	01/02/11	31/07/11

Tabla 48. Planificación de Tareas

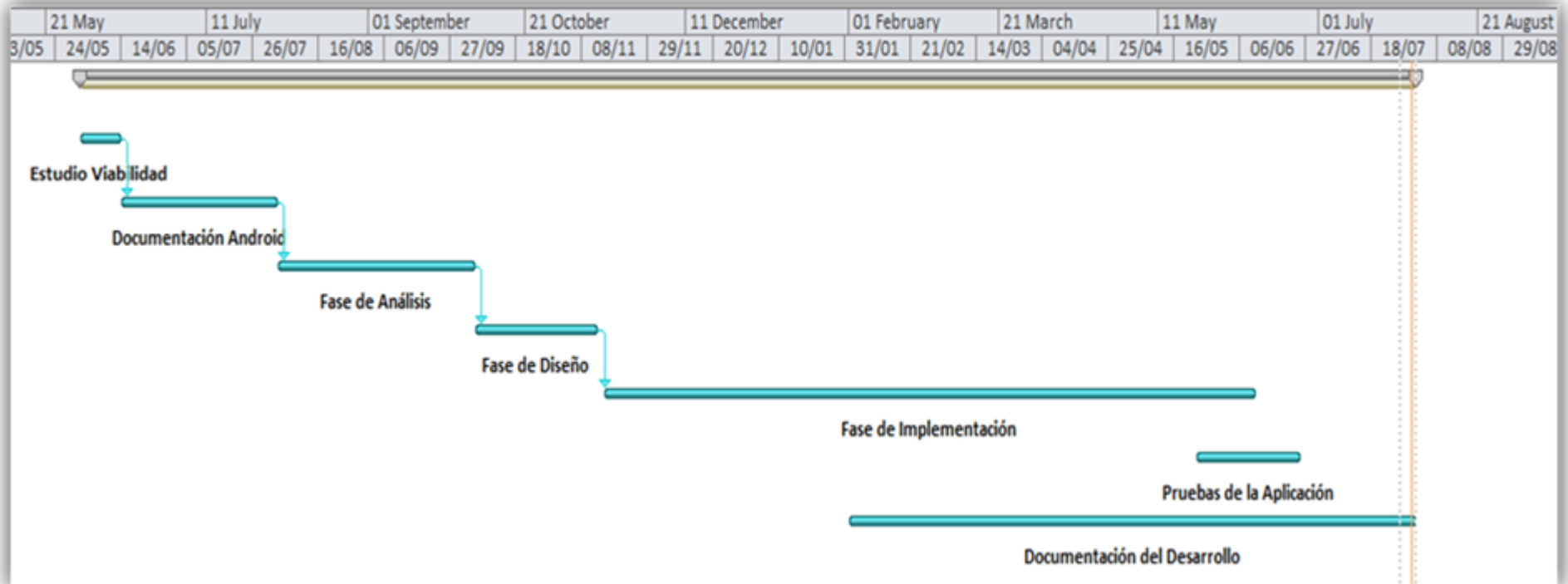


Figura 70. Diagrama de Gantt respecto a la Planificación del Proyecto

## 5.2 Presupuesto

Una vez establecida la duración y planificación que tendrá el proyecto, en esta sección se dispone a presentar el presupuesto para la realización del mismo. Para ello se ha utilizado como guía la plantilla proporcionada por la universidad Carlos III de Madrid [Plantilla presupuesto UC3M].

Tomando dicha plantilla como referencia, se desglosarán los gastos empezando por los personales. Éstos se calcularán tomando la planificación realizada en el apartado anterior pero tomando las siguientes consideraciones:

- ❖ El número medio de horas diarias dedicadas al proyecto han sido 3 horas, ya que como se ha mencionado anteriormente, no se ha realizado una dedicación total al desarrollo del proyecto al haberse compatibilizado su desarrollo con diversas tareas.
- ❖ Los días festivos que no se han dedicado al trabajo del mismo han sido:
  - 12 de Octubre 2010
  - 9 de Noviembre de 2010
  - 6, 8, 24 de Diciembre de 2010
  - 5 y 6 de Enero de 2011
  - 19 de Marzo de 2011
  - 21, 22, 23 y 24 de Abril de 2011
  - 2 de Mayo de 2011
  - 23 de Junio de 2011
  - 25 de Julio de 2011
- ❖ Las vacaciones tomadas durante el desarrollo del proyecto han sido:
  - 1 al 15 de agosto de 2010
  - 7 de Diciembre de 2010
  - 18, 19 y 20 de Abril de 2011
  - 15 al 31 de Mayo de 2011

Viendo estos datos, se calcula la duración real asociada a cada actividad dentro del proyecto quedando la nueva planificación como muestra la siguiente tabla.

<b>Tarea</b>	<b>Duración Real</b>
Estudio Viabilidad	10 días
Documentación Android	34 días
Fase de Análisis	32 días
Fase de Diseño	27 días
Fase de Implementación	133 días
Pruebas de la Aplicación	25 días
Documentación del Desarrollo	115 días

**Tabla 49. Duración real por tarea**

De esta manera se calcula el coste personal siguiendo la plantilla mencionada aplicando la fórmula de

$$\text{Coste} = ((\text{duración días} * \text{horas diarias}) / \text{dedicación hombre mes}) * \text{coste hombre mes}$$

Y sabiendo que

$$\text{duración días} = 376 \text{ días}$$

$$\text{horas diarias} = 3 \text{ horas}$$

$$\text{dedicación hombre mes} = 131,25 \text{ horas}$$

$$\text{coste hombre mes} = 2.694,39 \text{ €}$$

El coste personal del proyecto asciende a un total de VEINTITRÉS MIL CIENTO CINCUENTA Y SEIS EUROS CON TREINTA Y CINCO CÉNTIMOS DE EURO.

Por lo que a los costes de equipo se refiere, se ha adquirido como equipo un portátil ASUS 1201 N con un coste de 350 € sin IVA para trabajar en el desarrollo del proyecto. Siguiendo la plantilla para el cálculo de presupuestos, se debe calcular la amortización para el equipo.

Para ello se utilizará la siguiente formula

Dónde **A** = nº de meses desde la fecha de facturación en que el equipo es utilizado.

$$\frac{A}{B} \times C \times D$$

**B** = periodo de depreciación (60 meses)

**C** = coste del equipo (sin IVA)

**D** = % del uso que se dedica al proyecto (habitualmente 100%)

En la siguiente tabla se puede ver el coste imputable de amortización del equipo utilizado para el desarrollo del proyecto.

Descripción	Coste (Euro)	% Uso dedicado proyecto	Dedicación (meses)	Periodo de depreciación	Coste Imputable
ASUS 1201 N	350 €	100%	13	60	32,5 €

Tabla 50. Costes de amortización

A demás de los costes anteriores, hay que sumar los costes de las licencias para el desarrollo del proyecto. Estas licencias son para:

- ❖ Microsoft Office 2010: 379,00 €
- ❖ Microsoft Project 2010: 364,50 €

El total de estos costes asciende a SETECIENTOS CUARENTA Y TRES EUROS CON CINCUENTA CÉNTIMOS DE EURO.

Establecidos todos los costes, la siguiente tabla resume cada uno de ellos, estableciendo el coste total para el proyecto.

Descripción Presupuesto costes totales	Presupuesto Costes Totales
Costes Personales	23156,35 €
Costes Amortización	32,5 €
Costes Funcionales	743,50 €
Costes Indirectos (20%)	4786,47 €
Total sin IVA	28718,82 €
<b>Total con IVA (18%)</b>	<b>33888,20 €</b>

Tabla 51. Resumen presupuesto

El presupuesto total del proyecto asciende a la cantidad de **TREINTA Y TRES MIL OCHOCIENTOS OCHENTA Y OCHO EUROS CON VEINTE CÉNTIMOS DE EURO.**

Leganés, a 3 de Octubre de 2011,

El ingeniero proyectista



Fdo. Óscar Magro Segura.



# Capítulo 6

## Conclusiones y líneas futuras

### 6.1 Conclusiones

El desarrollo de este proyecto no ha sido fácil debido a que se ha tenido que superar una serie de retos y tomar diferentes decisiones para completar su desarrollo. Lo único que se tenía claro desde un principio era el tipo de proyecto a realizar, que sería el desarrollo de una aplicación para un dispositivo móvil.

Con esa idea, surgió la primera pregunta ¿sobre qué plataforma se iba a trabajar? Tras estudiar las diferentes plataformas del mercado y analizar sus ventajas y desventajas se optó por trabajar para Android. Esto origino el primer reto y uno de los más importantes, conocer la arquitectura de la plataforma Android y cómo se debe trabajar con ella, ya que era totalmente desconocida para el desarrollador a excepción de su lenguaje de programación lo que facilitó en gran medida el trabajo de aprendizaje.

Después vino la que puede haber sido la decisión más difícil de este desarrollo, ¿qué aplicación se iba a desarrollar? Existen numerosas aplicaciones muy distintas en el mercado por lo que idear una nueva aplicación desde cero no es algo fácil. Para facilitar la elección de desarrollo se plantearon dos preguntas, ¿qué es lo que necesito? y ¿qué es lo que quiero que haga mi aplicación? De estas dos premisas sale la idea de crear una aplicación que permita gestionar una de las cosas más valiosas que existen, el tiempo. Se necesitaba algo que permitiera organizar al usuario su tiempo, recordar posibles eventos y actividades.

Con esta idea se empezó a diseñar la aplicación y se pensó en posibles herramientas que facilitaran esta organización, por lo que al final se terminó interaccionando con la herramienta que más usuarios utilizan en internet, el correo electrónico y con el Calendario de Google.

El esfuerzo mayor de todo el desarrollo fue el de integrar la aplicación con los servicios de Google ya que fue bastante costoso comprender cómo funciona su protocolo y cómo adaptarlo para un desarrollo Android, pero una vez integrado, se obtenía la principal funcionalidad de la aplicación por lo que quedó demostrada la utilidad y viabilidad de la misma.

Desde un punto de vista personal, ver como con los conocimientos adquiridos durante la carrera en las diversas asignaturas ha permitido desarrollar una aplicación con una verdadera utilidad ha supuesto una alta satisfacción propia. Si a esto sumamos los nuevos conocimientos adquiridos en la etapa de autoaprendizaje a lo largo del desarrollo de este proyecto se ve cómo se cumplen con creces los objetivos propuestos y se amplían conocimientos a la vez que se plasman los que ya se poseían.

Desde el punto de vista del desarrollo, a lo largo de este proyecto se ha visto como el mercado de aplicaciones para dispositivos móviles está en pleno auge y es un sector fácil de explotar donde a partir de un coste económico bajo y medio si hablamos de esfuerzo, se pueden obtener unos altos beneficios.

Esto permite pensar si podría ser factible en un futuro no muy lejano el poder emprender una empresa y poder dedicarse a este sector el cual aún es muy joven y puede dar mucho de sí.

## 6.2 Líneas futuras

Después de haber finalizado el proyecto, existen diferentes mejoras y nuevos desarrollos que podrían llevarse a cabo en un futuro. A continuación se detallan una serie de líneas de trabajo que serían interesantes de estudiar para ver si podría ser factible su realización.

Respecto al componente del gestor de correo se propone:

- ❖ Generalizar el uso de servidores de correo y permitir al usuario acceder a diferentes servidores como MSN, o Yahoo entre otros, ya que actualmente se ha implementado para cuentas en Gmail. Para ello sería necesario que dichos servidores tuvieran habilitado el puerto de IMAP, ya que para el caso de MSN éste aún no lo está aunque no tardarán en incluirlo.

Respecto al componente del gestor de eventos se propone:

- ❖ Permitir al usuario que cree sus propias plantillas para los eventos de tal manera que pueda compartirlas con sus contactos. Podría definir una estructura y la propia aplicación se adaptaría a ella.
- ❖ Ampliar la interacción del usuario con su calendario ofreciendo la posibilidad de crear nuevos calendarios desde esta herramienta así como permitirle seleccionar cualquiera de los calendarios que posee para almacenar os eventos.

Por lo que a la seguridad de la autenticación en la aplicación se refiere se propone:

- ❖ Cambiar el tipo de autenticación de ClientLogin a AccountManager, lo que ofrecerá mayor confianza al usuario de la aplicación ya que ésta dejará de necesitar las credenciales del usuario al utilizar este gestor que ofrece Android que utiliza las cuentas definidas por el usuario en el terminal.

- [1] Por último, durante la fase de análisis al inicio del proyecto, se estudió añadir una funcionalidad adicional a la aplicación que consistía en utilizar la tecnología GPS del dispositivo para que cuando el usuario estuviera cerca de una localización en el que se había definido un evento, mediante el GPS lanzar un aviso al usuario indicando el recordatorio de dicha actividad y le indicara cómo llegar hasta el lugar. Este desarrollo decidió dejarse a parte debido a que aumentaba la complejidad del proyecto por lo que se pensó en proponerlo como línea futura de desarrollo. Como dato informativo, Apple publicó el 2 de Junio de 2011 una patente de un software que realizaba algo parecido. Investigación de Apple de localización dinámica para alertas de calendario  
[http://www.appleinsider.com/articles/11/06/02/apple\\_investigating\\_dynamic\\_location\\_\\_traffic\\_aware\\_iphone\\_calendar\\_alerts.html](http://www.appleinsider.com/articles/11/06/02/apple_investigating_dynamic_location__traffic_aware_iphone_calendar_alerts.html)

- [2] Ventas smartphone frente a móviles tradicionales:

[http://www.elpais.com/articulo/tecnologia/Europa/venden/smartphones/moviles/elpeputec/20110913elpeputec\\_3/Tes](http://www.elpais.com/articulo/tecnologia/Europa/venden/smartphones/moviles/elpeputec/20110913elpeputec_3/Tes)

, y como se puede ver por la fecha, esta publicación fue posterior a la idea que surgió durante la definición de funcionalidades de la aplicación desarrollada.

# Apéndice a

## Plataforma Android

### a.1. Introducción

La intención de este apéndice es describir en detalle el sistema operativo Android para aquellos que desconozcan sus características y funcionamiento.

Como ya se ha comentado al principio del documento, Android es un sistema operativo móvil desarrollado inicialmente por Android Inc. y en 2005 fue adquirido por Google.

Esta plataforma se basa en la versión 2.6 de Linux para el sistema de servicios básicos, tales como la seguridad, la gestión de memoria, la gestión de recursos, la pila de red y el modelo del controlador. A su vez, su núcleo también actúa como una capa de abstracción entre el hardware y el resto de la pila de software.

A lo largo de esta sección se explicarán más detalles de este sistema operativo que ayudarán a comprender sus características.

## a.2. Versiones

Desde que apareció el sistema operativo en Septiembre de 2008, Android ha pasado por diferentes revisiones mejorando su rendimiento con la intención de corregir errores y añadir nuevas características, producto de lo cual son las diferentes versiones de la plataforma. Este historial de versiones contiene:

### ❖ Android 1.0

Se trata de la primera versión de Android que apareció en el mercado y fue lanzada el 23 de Septiembre de 2008.

### ❖ Android 1.1

Apareció el 9 de Febrero de 2009 con la intención de actualizar la plataforma únicamente para el terminal HTC Dream o también conocido como T-Mobile G1. Estas mejoras incluían:

- Resolución de fallos.
- Cambios en el API.
- Agregar detalles y cambios en los mapas.
- Tiempo de espera más largo para la pantalla cuando se utilizaba el altavoz.
- La capacidad de ocultar y mostrar el teclado de marcación presente en el menú de llamada.
- Soporte para almacenar adjuntos de los mensajes multimedia.
- Soporte para la capa de diseño.

### ❖ Android 1.5 (Cupcake)

Esta versión fue bastante importante ya que no solo arreglaba algunos fallos si no que incluía numerosas mejoras, por lo que directamente se saltó de la 1.1 a la 1.5. Apareció el 30 de Abril de 2009 y algunas de estas mejoras fueron:

- Nueva funcionalidad para grabar y reproducir vídeos.
- Permitía la posibilidad de subir videos a YouTube al igual que subir fotos al directorio de Picasa desde el mismo dispositivo.
- Nuevo software de predicción de texto para el teclado así como la posibilidad de cambiar la orientación del teclado.
- Transiciones animadas entre pantallas.
- Nuevos widgets y carpetas que permitían personalizar la interfaz principal.
- Incorporación de soporte para Bluetooth A2DP y AVRCP.
- Habilidad para conectar automáticamente a dispositivos Bluetooth desde una cierta distancia.

❖ Android 1.6 (Donut)

El 15 de Septiembre de 2009 se lanzó esta nueva versión basándose en el Kernel de Linux 2.6.29. A pesar de no tratarse de una actualización grande, los cambios que introdujo mejoraron bastante las funcionalidades.

- Mejora del Android Market con respecto a la organización e información de las aplicaciones.
- Mejoras en la interfaz de galería, permitiendo por ejemplo selecciones múltiples.
- Actualización en la búsqueda por voz, aumentando su velocidad de respuesta.
- Aparece un nuevo motor de búsqueda en la pantalla de inicio el cual permite buscar en diferentes fuentes como contactos, historial, navegador y Google. A demás incluye funciones de autocompletado.
- Nuevo soporte para resoluciones de pantalla.
- Inclusión de soporte para conexiones a redes VPN, 802.1x.
- Aumento de velocidad en las aplicaciones de la cámara.

❖ Android 2.0

Apareció en Noviembre de 2009 introduciendo importantes mejoras tales como:

- Bluetooth 2.1.
- Mejora de Google Maps permitiendo soportar capas de versionado y la funcionalidad multi-táctil.
- Mejoras en la interfaz de usuario.
- Nuevo navegador permitiendo soporte para HTML5.
- Mejoras en la cámara como soporte de flash, efectos de color, modo escena y zoom digital.
- Soporte para nuevas resoluciones y tamaños de pantalla.
- Rediseño de la lista de contactos.
- Soporte para aplicaciones de redes sociales.
- Soporte de Microsoft Exchange.
- Mejora del teclado virtual.

❖ Android 2.1 (Eclair)

Se trata de una versión menor lanzada en Enero de 2010 con el fin de incluir cambios en el API y corregir algunos errores. Las principales características de esta versión fueron:

- Reconocimiento de voz.
- Mejoras en la galería tales como la inclusión de imágenes 3D.
- Mejoras de rendimiento para aumentar la duración de la batería.
- Mejoras en Google Maps permitiendo autocompletado en búsquedas y modo noche.

❖ Android 2.2 (Froyo)

Fue lanzada en Mayo de 2010 e incluía características tanto orientadas al usuario como al desarrollador. Aunque fue una versión menor, las mejoras fueron importantes. Algunas de ellas son:

- Optimizaciones en la velocidad del sistema operativo y memoria.
- Integración en el navegador Chrome el motor JavaScript V8.
- Posibilidad de utilizar el dispositivo como modem utilizando la conexión USB así como de punto de acceso Wifi.
- Actualizaciones automáticas para las aplicaciones.
- Soporte Wifi 802.11n
- Soporte para Radio FM.
- Soporte Flash 10.1 y Adobe AIR 2.5.
- Soporte para resoluciones de pantalla en alta definición.

❖ Android 2.3 (Gingerbread)

Se trata de la versión más actual en el momento para terminales móviles. Apareció en Diciembre de 2010. Las mejoras más importantes son:

- Actualización del diseño de la interfaz de usuario.
- Soporte nativo para telefonía VOIP.
- Soporte para pantallas extra grandes y resoluciones WXGA y mayores.
- Soporte para reproducción de videos WebM/VP8 y decodificación de audio AAC.
- Nuevos efectos de audio como reverberación, ecualización, virtualización de los auriculares y refuerzo de graves.
- Mejoras en la entrada de datos, audio y gráficos para desarrolladores de juegos.
- Administrador de descargas para archivos de gran tamaño.
- Soporte para múltiples cámaras.

❖ Android 3.0 (Honeycomb)

Se trata de una versión orientada a las tabletas y fue lanzada el 22 de Febrero de 2011. Los principales cambios que incluye son:

- Escritorio 3D con mejora de widgets.
- Mejora del sistema multitarea.
- Mejora en el navegador web predeterminado, incorporando pestañas.
- Soporte para video chat mediante Google Talk.



## a.3. Arquitectura

La siguiente figura muestra los diferentes niveles que se pueden encontrar en la arquitectura del Android.

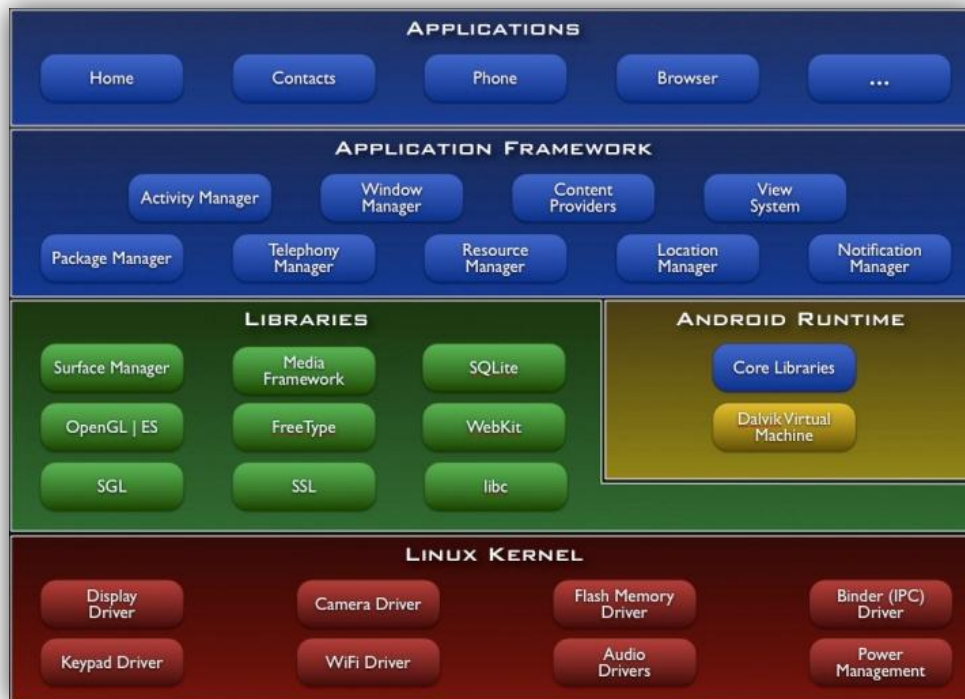


Figura 71. Arquitectura de Android

### a.3.1. Nivel de aplicaciones

Se corresponde al nivel de las aplicaciones base de la plataforma. Éstas incluyen un cliente de correo electrónico, un programa para mandar y gestionar SMS, un calendario, mapas y navegador web y una aplicación de gestión de contactos entre otras. Todas las aplicaciones de este nivel están programadas mediante el lenguaje de programación Java utilizando el SDK proporcionado por Android.

### a.3.2. Marco de trabajo de aplicaciones

Con el fin de facilitar la tarea a los desarrolladores, Android ofrece un framework mediante el cual permite acceder y utilizar cualquier dispositivo presente en el teléfono móvil para poder desarrollar aplicaciones. De esta manera se pueden acceder a los servicios de los distintos componentes y reutilizar sus funcionalidades.

### **a.3.3. Librerías de Android**

Android dispone de un conjunto de librerías muy extenso. Por un lado se encuentra un grupo de librerías base, escritas en Java, disponible en el paquete del SDK ofreciendo funcionalidades que serán utilizadas por las aplicaciones en su tiempo de ejecución.

A parte de las mencionadas, también se pueden encontrar otro conjunto de bibliotecas escritas en C/C++ que son usadas por los componentes del sistema y que mediante el marco de trabajo pueden ser utilizadas por las aplicaciones. Algunas de ellas son System C Library, bibliotecas de medios, bibliotecas gráficos y bibliotecas de SQLite.

### **a.3.4. Runtime de Android**

En este nivel se encuentra la máquina virtual Dalvik. Esta máquina virtual se caracteriza por tener una arquitectura basada en registros a diferencia de otras máquinas virtuales que poseen una arquitectura basada en pila. De esta manera cada aplicación de Android ejecuta su propio proceso con una instancia de la máquina virtual. Está diseñada para ocupar poco espacio de memoria y no requerir una velocidad alta de procesamiento.

Otra diferencia es que las clases que componen una aplicación son convertidas, utilizando una herramienta incluida en el SDK denominada dx. Así mismo, desde la versión 2.2 de Android, esta máquina virtual incluye un compilador JIT (Just in Time) aumentando el rendimiento de forma notable en las aplicaciones.

### **a.3.5. Kernel de Linux**

Como ya se ha comentado anteriormente, Android está basado en el Kernel de Linux 2.6 actuando como una capa de abstracción entre el hardware y el resto de capas de la arquitectura. Las funciones más significativas que cumple son de seguridad, gestión de memoria y procesos, protocolo de red y gestión de drivers del sistema.

## a.4. Aplicaciones

Las ejecuciones de aplicaciones en Android se basan en “*el principio del privilegio mínimo*” en el cual cada aplicación, por defecto solo tiene acceso únicamente a los recursos que necesita para llevar a cabo su tarea. De esta manera se obtiene un entorno seguro en el cual una aplicación no puede acceder a partes del sistema que no le han sido permitidas.

Para conseguir esto, Android se basa en un sistema operativo de Linux en el cual cada aplicación es tomada como un usuario diferente. Cuando se inicia la aplicación, el sistema le asigna un identificador único con una serie de permisos para todos los archivos de la aplicación, de manera que únicamente el usuario con ese identificador pueda acceder a ellos. A su vez, cada aplicación se ejecuta por defecto en un único proceso. Este proceso tiene su propia máquina virtual de tal manera que el código de la aplicación se ejecuta aislado. Esta ejecución se inicia cuando cualquiera de los componentes de la aplicación necesita ser lanzado y cuando deja de ser utilizado o el sistema necesita recuperar memoria para otras aplicaciones el proceso es cerrado.

También puede haber ocasiones en las que se dos aplicaciones distintas necesiten compartir datos o recursos o que una aplicación deba acceder a servicios del sistema. En el caso que dos aplicaciones necesiten compartir datos y no se rompa el principio descrito, Android permite que dos aplicaciones puedan compartir el mismo identificador, lo que hace que tengan acceso entre ellas. Para conservar los recursos del sistema, las aplicaciones con el mismo identificador pueden ser ejecutadas en el mismo proceso y en la misma máquina virtual. Por otro lado, para el caso que una aplicación necesite acceder a recursos del sistema tales como datos de contactos, conexión a internet u otros, estos permisos deben ser concedidos por el usuario a la hora de la instalación de la aplicación.

Una vez explicado el comportamiento de ejecución de una aplicación se describirán los cuatro componentes de los cuales una aplicación puede constar.

❖ Actividades

Una actividad es un componente que proporciona una ventana con la cual los usuarios pueden interaccionar. Una aplicación puede consistir de una única actividad o, como es más común, de múltiples actividades que se relacionan entre ellas. Un ejemplo sencillo puede ser el listado de los contactos telefónicos, el cual permite al usuario interaccionar con la lista y seleccionar un registro para o bien realizar llamadas o enviar algún mensaje.

En estos casos en los que una aplicación posee distintas actividades independientes, por lo general, una actividad es seleccionada como principal, y es la que se presenta al usuario al lanzar la aplicación. Cada actividad puede llamar a otras actividades para realizar distintas acciones, por lo que la actividad inicial queda suspendida, siendo el sistema el que la mantiene en la pila. La pila sigue la lógica de “*last in, first out*”, es decir que cuando una actividad se inicia, se coloca en la pila y toma el foco de atención mostrándose al usuario y ejecutando su lógica. Cuando el usuario acaba con la actividad y presiona la tecla “*back*”, la actividad se saca de la pila destruyéndose y la anterior actividad se reanuda.

Cada vez que se realiza un cambio de estado en una actividad se notifica mediante los métodos del ciclo de vida. Existen tres estados posibles por los que una actividad puede pasar. Cuando la actividad se encuentra en primer plano y se está ejecutando se encuentra en estado *activada* o en *reanudación*. Si pierde el foco y otra actividad está en primer plano pero ésta aún es visible la actividad se encuentra en el estado de *pausa*. Si por el contrario la actividad queda totalmente oculta pasa al estado de *parada*. En cualquiera de los tres estados, la actividad aún sigue presente en la pila, pero si una actividad está pausada o detenida el sistema puede terminar su ejecución liberando la memoria, ya sea bien porque la actividad termina o porque necesita terminarla.

Como se ha mencionado, para realizar la transición entre los diferentes estados existen una serie de métodos que son llamados cada vez que se quiere modificar el ciclo de vida de la aplicación. Éstos son *onCreate()*, *onStart()*, *onRestart()*, *onResume()*, *onPause()*, *onStop()* y *onDestroy()*. Implementando estos métodos se pueden monitorizar tres bucles anidados en el ciclo de vida.

## ○ Ciclo de vida completo

La actividad toma lugar entre la llamada de creación *onCreate()* y destrucción *onDestroy()*. En este caso la actividad debe llevar a cabo su configuración y tomar los recursos necesarios en su creación y liberarlos en su destrucción.

- Tiempo de vida visible

Este tiempo tiene lugar entre la llamada *onStart()* y la llamada *onStop()*. Durante este periodo el usuario puede ver la actividad aunque no esté en primer plano y ésta puede mantener sus recursos para poder ser mostrada. El sistema puede llamar a estas funciones varias veces a lo largo del ciclo de vida de la aplicación mientras la actividad se alterna entre visible y oculta para el usuario.

- Tiempo de vida en primer plano

Ocurre cuando la actividad se encuentra entre la llamada *onResume()* y *onPause()*. Durante este tiempo la actividad está en primer plano y el usuario interactúa con ella.

La siguiente figura muestra gráficamente el ciclo de vida de una actividad así como los bucles descritos anteriormente. En la imagen se representan los estados y las llamadas así como sus transiciones.

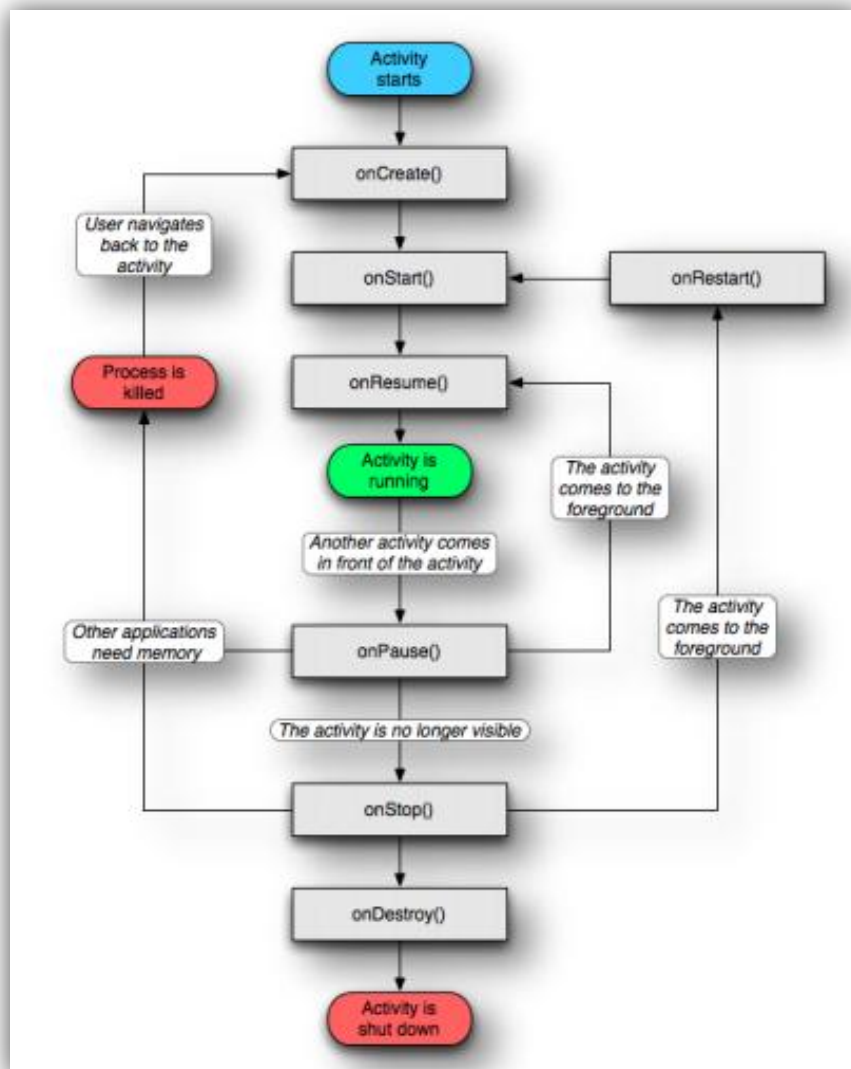


Figura 72. Ciclo de vida de una actividad

❖ Servicios

Los servicios son componentes que a diferencia de las actividades, efectúan operaciones en segundo plano y no utilizan interfaces de usuario. Son ejecuciones que se inician en un momento determinado, ya sea por una llamada de una actividad, y que se paran o bien cuando acaban su función, o bien cuando son detenidos. El ejemplo más claro es el servicio que permite sincronizar el buzón de correo. Por defecto comienza al terminar de iniciarse el sistema operativo y mientras haya conexión a internet se irá ejecutando para poder sincronizar el buzón de correo con el servidor, pero el usuario siempre puede detenerlo cuando desee.

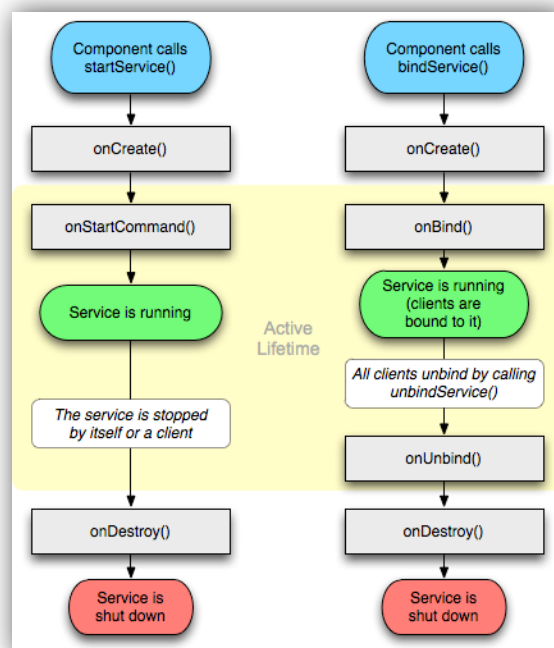
Al igual que las actividades, los servicios poseen un ciclo de vida que viene determinado por los métodos *onCreate()*, *onStart()*, *onBind()*, *onUnbind()* y *onDestroy()*. Dependiendo de su implementación pueden darse dos bucles de ejecución:

- Ciclo de vida activo

El servicio es iniciado con la llamada a *onStart()* y se ejecuta indefinidamente hasta que o bien se para por sí mismo u otro componente se encarga de finalizarlo. Cuando el servicio termina el propio sistema se encarga de destruirlo y liberar sus recursos.

- Ciclo de vida ligado

El servicio es creado cuando otro componente, cliente, hace la llamada al método *onBind()*. El cliente se comunica a través de la interfaz *IBinder* y puede cerrar la conexión con *onUnbind()*. Múltiples clientes pueden ligarse al mismo servicio y cuando todos se desligan, el servicio es destruido por el sistema, pero a diferencia del anterior ciclo, el servicio no puede destruirse a sí mismo.



**Figura 73. Ciclos de vida de un servicio**

❖ Proveedores de contenido

Permiten gestionar y compartir datos en las aplicaciones. Lo más común es utilizar el almacenamiento de datos proporcionado por SQLite, aunque puede usarse cualquier otro medio de almacenamiento que pueda ser accedido por la aplicación. A través de este gestor de contenido, las aplicaciones pueden acceder a los datos bien para consultar o modificar la información almacenada.

❖ Receptores de difusión

Este componente se encarga de recibir y responder ante mensajes. La mayoría de éstos son originados en el sistema, como por ejemplo cuando se apaga la pantalla., aunque las aplicaciones pueden iniciarlos con el fin de avisar al usuario.

En lugar de mostrar una interfaz gráfica, suelen crear un estado en la barra de notificaciones para avisar al usuario.

El ciclo de vida de estos componentes es un único método llamado *onReceive*. Este método se ejecuta cuando llega un mensaje, pasándose por parámetro, y el receptor se considera que está activo, por lo que un proceso que contenga un receptor de difusión activo no podrá ser eliminado de la memoria del sistema.

## a.5. Proveedores de contenido

Los proveedores de contenido tienen como finalidad almacenar los datos y hacerlos accesibles para todas las aplicaciones. Son la única forma de compartir los datos entre distintas aplicaciones ya que Android no posee un área de intercambio de datos.

Android incluye una serie de proveedores de contenido para tipos de datos comunes como por ejemplo audio, vídeo, imágenes e información de contactos. Aunque esto permite compartir esa información con las aplicaciones, en algunos casos es necesario obtener ciertos permisos para acceder a los datos.

A la hora de compartir los datos hay dos opciones, o bien el desarrollador crea un proveedor de contenido propio, extendiendo de la clase *ContentProvider* o puede añadir los datos a un contenedor existente siempre y cuando disponga de permisos para escribir en él y esté manejando el mismo tipo de datos.

A pesar de cómo sea la forma que esté creado el contenedor, para obtener los datos se hace mediante consultas usando una interfaz común y generalmente usando objetos del tipo *ContentResolver* que devuelven un cursor que permite manejar los datos. De esta manera los desarrolladores pueden manejar datos entre aplicaciones con gran facilidad.

## a.6. Interfaz de Usuario

En una aplicación Android, la interfaz de usuario está creada mediante los componentes *View* y *ViewGroup*, aunque existen diferentes tipos de vistas y grupos y cada uno de ellos heredan de la clase *View*.

Un elemento *View* es la unidad básica de la interfaz de usuario que almacena la información de la disposición en pantalla y el contenido de las regiones. A su vez, estos objetos constituyen una base para otros elementos llamados *widgets* con funciones ya implementadas que pueden ser utilizados en la interfaz de usuario. Por otra parte, los objetos *ViewGroup* son la base para los *layouts* que ofrecen diferentes arquitecturas y disposiciones como pueden ser lineales, en tablas o relativos.

En la plataforma Android, cuando se define una interfaz de usuario, se utiliza una jerarquía de nodos *View* y *ViewGroup* en forma de árbol. Esta jerarquía puede ser tan sencilla como compleja, dependiendo de las necesidades y de los elementos que se vayan a añadir. La siguiente figura muestra este tipo de jerarquías.

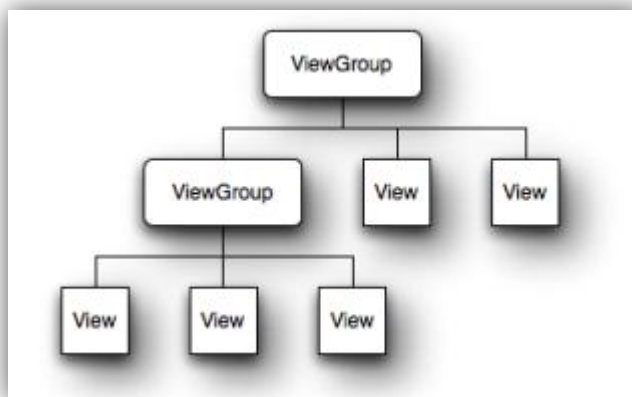


Figura 74. Jerarquía de interfaz de usuario

La forma más común de definir y expresar la jerarquía mencionada es usando archivos XML. Cada elemento contenido en estos archivos es o bien un objeto del tipo *View* o *ViewGroup* o bien de objetos que heredan de éstos, siendo los nodos hoja del árbol de esta jerarquía los objetos *View* y las ramas del árbol los objetos *ViewGroup*. Cada elemento tiene un nombre distintivo que corresponde a la clase Java que representa. Así pues un elemento `<TextView>` crea un *TextView* en la interfaz de usuario y cuando se cargan los recursos de la interfaz, el sistema Android inicializa los objetos correspondientes a los elementos de la interfaz.



A continuación se muestra un ejemplo simple en el cual se utiliza un *ViewGroup* lineal, llamado *LinearLayout*, que contiene dos elementos *View*, un texto y un botón, que se mostrarán uno debajo del otro al tener las propiedades de orientación configuradas a vertical.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical" android:background="@drawable/backwhite">
    <TextView android:id="@+id/text"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a TextView" />
    <Button android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello, I am a Button" />
</LinearLayout>
```

Figura 75. Código de ejemplo de una interfaz de usuario

La salida de este fragmento de código originará la siguiente interfaz:



Figura 76. Salida de ejemplo de una interfaz de usuario

Por último hay que comentar que también existe otra parte importante de la interfaz en una aplicación y son los menús. Éstos ofrecen una interfaz sencilla que permite realizar algún cambio o configuración dentro de la aplicación. La forma más común de usarlos es mediante el botón ya incorporado en el terminal móvil, aunque también existe el uso de menús contextuales que son mostrados cuando el usuario mantiene apretado un elemento de la interfaz.

Estos menús también se organizan estructurándose en una jerarquía como la descrita anteriormente pero en lugar de definirse de la misma forma, se usa una serie de funciones y llamadas, como *onCreateContextMenu()* y *onCreateOptionsMenu()*, desde dentro de la actividad permitiendo definir los elementos que se quieren incluir en los menús y el sistema se encarga de crear la estructura.

Por lo tanto, gracias a las funcionalidades que ofrece Android a la hora de trabajar con las interfaces de una aplicación, es posible personalizarlas totalmente permitiendo aplicar estilos y formas visuales.

## a.7. Recursos de Aplicación

A la hora de utilizar los recursos de una aplicación como imágenes o cadenas de texto, una buena práctica es externalizarlos y referenciarlos, permitiendo que su mantenimiento permanezca independiente. Esto también permite a su vez definir recursos alternativos para diferentes configuraciones de dispositivos, como por ejemplo crear aplicaciones multilenguaje o ajustar las interfaces a diferentes tamaños de pantalla. Para lograr esa compatibilidad con distintas configuraciones se deben organizar los recursos en los proyectos almacenándolos en el directorio *res/* usando sub-carpetas para agrupar los recursos dependiendo de la configuración.

Para cualquier tipo de recurso se puede definir un valor por defecto y múltiples valores alternativos. Los valores por defecto son aquellos que deben ser usados independientemente del tipo de configuración del dispositivo o cuando no hay recursos alternativos que se ajusten a un tipo de configuración. Los alternativos son aquellos que se utilizarán para las configuraciones que han sido diseñadas previamente.

Para verlo más claro, un ejemplo de utilización de recursos alternativos puede ser cuando dependiendo de la orientación de la pantalla se desea mostrar una interfaz diferente. Para ello se deben definir dos interfaces distintas, una para cuando el dispositivo se encuentra en modo vertical y otra para cuando esté en horizontal, ya que se desea mostrar la interfaz apaisada y automáticamente el sistema tomará la interfaz que corresponde. A continuación se muestra gráficamente el ejemplo para dos dispositivos con diferentes tipos de pantalla. Para la primera imagen no existe una alternativa de interfaz por lo que ambos dispositivos muestran el mismo recurso, provocando que el dispositivo B, al tener una pantalla horizontal no muestre correctamente la interfaz desaprovechando espacio. En la segunda imagen, al proveer a la aplicación una segunda interfaz, el dispositivo B utilizará el recurso alternativo para mostrar de forma correcta la interfaz de usuario.

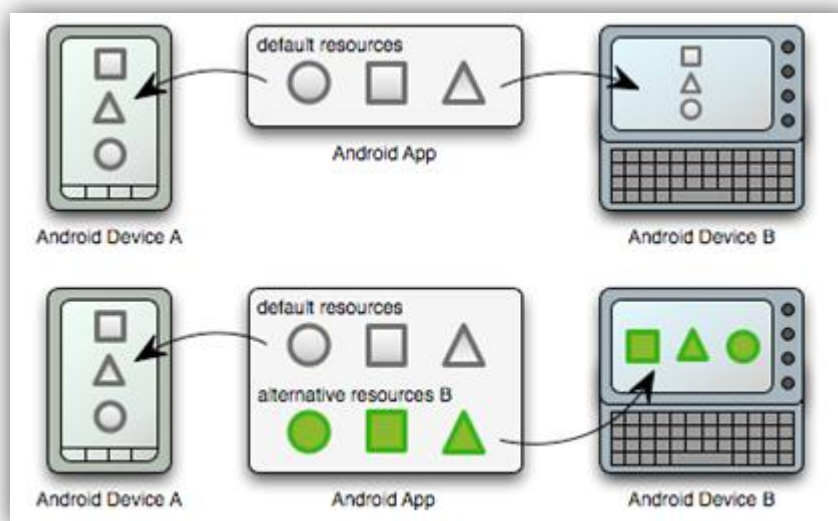


Figura 77. Uso de recursos alternativos

## a.8. Seguridad y permisos

Como ya se ha explicado en el apartado de las aplicaciones en Android, éste es un sistema multi-tarea que aplica una política de privilegios separados en el que cada aplicación se ejecuta con un identificador único. A su vez también se separan las partes del sistema en diferentes identidades lo que permite que el Kernel de Linux logre un aislamiento de las aplicaciones entre ellas mismas y el sistema.

Android provee otro tipo de características que permiten mantener un alto nivel de seguridad basadas en mecanismos de permisos, los cuales imponen una serie de restricciones en las operaciones que un proceso puede realizar, y permisos de acceso add-hoc a los datos mediante mecanismos de URIs.

Si se enfoca la seguridad desde el punto de la arquitectura, Android establece que por defecto ninguna aplicación tiene permisos para realizar operaciones que puedan impactar sobre el resto de aplicaciones, el sistema operativo o al usuario. Esto incluye la lectura o escritura de los datos privados del usuario como sus contactos, o lectura y escritura de los archivos de otras aplicaciones, o acceso a internet entre otras acciones.

A pesar de lo mencionado, se dan ocasiones en las que las aplicaciones necesitan acceder a ciertos recursos o realizar una serie de tareas que no están permitidas por defecto. Para que Android pueda permitir esas acciones, se deben de indicar estos permisos en el fichero xml *AndroidManifest*. Para ello se incluirán una serie de entradas en el archivo mediante el uso de etiquetas indicando los permisos que se necesitan. En la siguiente imagen se muestra un ejemplo para el caso en el que la aplicación desee acceder a internet.

```
</application>
<uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

**Figura 78. Permiso acceso internet**

Así pues, no existe mucha complicación a la hora de añadir permisos mediante el fichero *AndroidManifest* a la aplicación. Pero hay veces que para permitir actividades a las aplicaciones no basta con los permisos explicados. Hay ocasiones en las que usando los proveedores de contenido hay que especificar los permisos usando definiciones de URIs para poder acceder a los datos. Un ejemplo para garantizar acceso de lectura mediante estas definiciones es utilizando *Intent.FLAG\_GRANT\_READ\_URI\_PERMISSION*.

Gracias a estos métodos se pueden mantener la seguridad en las aplicaciones y a su vez permitir que éstas puedan realizar las acciones que necesiten siempre y cuando sean aceptadas.

## a.9. Archivo AndroidManifest.xml

Este fichero debe estar siempre contenido en toda aplicación Android en su directorio raíz. El archivo *AndroidManifest.xml* contiene información esencial de la aplicación para el sistema operativo que debe tener en cuenta antes de que se lance la aplicación. Las características más importantes que debe contener son:

- ❖ El nombre del paquete Java de la aplicación ya que este nombre servirá como identificador único para la aplicación.
- ❖ La descripción de los componentes de la aplicación. Estos son las actividades, servicios, receptores y los proveedores de contenido que la aplicación necesitará, nombrando las clases que las implementan.
- ❖ Los procesos que contendrán los componentes descritos.
- ❖ Los permisos que la aplicación debe tener con el fin de poder acceder a las partes protegidas del API e interactuar con otras aplicaciones o componentes.
- ❖ El nivel mínimo del API de Android que la aplicación necesita.

La siguiente imagen muestra una estructura general que el archivo debe tener:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest>
  <uses-permission />
  <permission />
  <permission-tree />
  <permission-group />
  <instrumentation />
  <uses-sdk />
  <uses-configuration />
  <uses-feature />
  <supports-screens />
  <compatible-screens />
  <supports-gl-texture />
  <application>
    <activity>
      <intent-filter>
        <action />
        <category />
        <data />
      </intent-filter>
      <meta-data />
    </activity>
    <activity-alias>
      <intent-filter> . . . </intent-filter>
      <meta-data />
    </activity-alias>
    <service>
      <intent-filter> . . . </intent-filter>
      <meta-data />
    </service>
    <receiver>
      <intent-filter> . . . </intent-filter>
      <meta-data />
    </receiver>
    <provider>
      <grant-uri-permission />
      <meta-data />
    </provider>
    <uses-library />
  </application>
</manifest>
```

Figura 79. Estructura general del archivo AndroidManifest.xml

# Glosario

ADT	<i>Android Developers Tools</i>
APP	<i>Application</i>
AVD	<i>Android Virtual Machine</i>
GPS	<i>Global Positioning System</i>
IDE	<i>Integrated Development Environment</i>
JIT	<i>Just in Time</i>
MIME	<i>Multipurpose Internet Mail Extensions</i>
PDA	<i>Personal Digital Assistant</i>
SD	<i>Secure Digital</i>
SDK	<i>Software Development Kit</i>
SMS	<i>Short Message Service</i>
SSL	<i>Secure Socket Layer</i>
UI	<i>User Interface</i>
URI	<i>Uniform Resource Identifier</i>
USB	<i>Universal Serial Bus</i>
VPN	<i>Virtual Private Network</i>
XML	<i>Extended Markup Language</i>

# Referencias

- [3] Dispositivo móvil:  
[http://es.wikipedia.org/wiki/Dispositivo\\_m%C3%B3vil](http://es.wikipedia.org/wiki/Dispositivo_m%C3%B3vil)
  
- [4] Datos de estudio de mercado sobre líneas móviles de la comisión de mercado de las telecomunicaciones (CMT):  
<http://blogcmt.com/wp-content/uploads/2010/08/Evoluci%C3%B3n-de-l%C3%ADneas-m%C3%B3viles.png>  
[http://www.cmt.es/cmt\\_ptl\\_ext/SelectOption.do](http://www.cmt.es/cmt_ptl_ext/SelectOption.do)  
[http://www.cmt.es/en/publicaciones/anexos/0100511\\_NM\\_marzo010.pdf](http://www.cmt.es/en/publicaciones/anexos/0100511_NM_marzo010.pdf)
  
- [5] Evolución de los terminales móviles:  
<http://www.muymovil.com/2009/10/10/la-evolucion-de-los-telefonos-moviles>
  
- [6] Smartphones:  
<http://en.wikipedia.org/wiki/Smartphone>
  
- [7] Encuesta de Handmark sobre el uso de medios para acceder a las noticias por los usuarios:  
<http://mashable.com/2010/12/07/smartphones-breaking-news-study/>
  
- [8] Cuota de mercado de las plataformas móviles:  
<http://www.phonecurry.com/blog/symbian-vs-android-vs-windows-mobile-vs-blackberry-vs-iphone/>  
<http://nextbigfuture.com/2010/09/cellphone-os-forecast-for-2011-and-2014.html>  
[http://en.wikipedia.org/wiki/Mobile\\_operating\\_system](http://en.wikipedia.org/wiki/Mobile_operating_system)

- [9] Número de aplicaciones en “App Stores”:  
<http://www.phonecurry.com/blog/symbian-vs-android-vs-windows-mobile-vs-blackberry-vs-iphone/>
- [10] Comparativa Android vs IOS:  
<http://www.xatakamovil.com/galeria/big/tabla-comparativa-ios4-vs-froyo/1>  
[http://www.cnet.com/8301-17918\\_1-20002198-85.html](http://www.cnet.com/8301-17918_1-20002198-85.html)  
<http://www.wired.com/gadgetlab/2010/06/comparison-apple-versus-android/>  
<http://www.taranfx.com/iphone-4-vs-android-2-2>
- [11] Android:  
<http://developer.android.com/guide/basics/what-is-android.html>
- [12] Versiones Android:  
<http://developer.android.com/sdk/>
- [13] Protocolo GoogleCalendar  
[http://code.google.com/intl/es/apis/calendar/data/2.0/developers\\_guide\\_protocol.html](http://code.google.com/intl/es/apis/calendar/data/2.0/developers_guide_protocol.html)
- [14] IDE Eclipse  
<http://www.eclipse.org/>
- [15] SDK Android  
<http://developer.android.com/sdk/index.html>
- [16] Librería JavaMail de Oracle  
<http://www.oracle.com/technetwork/java/javamail/index.html>
- [17] Librerías JavaMail para Android  
<http://code.google.com/p/javamail-android/downloads/list>
- [18] Librerías GoogleCalendar para Android  
<http://code.google.com/p/google-api-java-client/wiki/Android>
- [19] Plantilla presupuesto UC3M  
[https://www.uc3m.es/portal/page/portal/administracion\\_campus\\_leganes\\_est\\_cg/proyecto\\_fin\\_carrera/Formulario\\_PresupuestoPFC-TFG%20\(3\)\\_2.xlsx](https://www.uc3m.es/portal/page/portal/administracion_campus_leganes_est_cg/proyecto_fin_carrera/Formulario_PresupuestoPFC-TFG%20(3)_2.xlsx)
- [20] Investigación de Apple de localización dinámica para alertas de calendario  
[http://www.appleinsider.com/articles/11/06/02/apple\\_investigating\\_dynamic\\_location\\_traffic\\_aware\\_iphone\\_calendar\\_alerts.html](http://www.appleinsider.com/articles/11/06/02/apple_investigating_dynamic_location_traffic_aware_iphone_calendar_alerts.html)
- [21] Ventas smartphone frente a móviles tradicionales:  
[http://www.elpais.com/articulo/tecnologia/Europa/venden/smartphones/moviles/elpeputec/20110913elpeputec\\_3/Tes](http://www.elpais.com/articulo/tecnologia/Europa/venden/smartphones/moviles/elpeputec/20110913elpeputec_3/Tes)